LECTURE NOTES

ON

E-COMMERCE

COURSE CODE: **58061**
BRANCH: **IT**

B.Tech - **IV YEAR II SEM**

Mr. RAHUL
ASSISTANT PROFESSOR

INFORMATION TECHNOLOGY

INSTITUTE OF AERONAUTICAL ENGINEERING
(Autonomous)
DUNDIGAL, HYDERABAD - 500 043

UNIT-1

E-Business Introduction

E-Business vs. E-commerce

While some use e-commerce and e-business interchangeably, they are distinct concepts. **Electronic business**, commonly referred to as "**e-Business**" or "**e-business**", may be defined as the application of information and communication technologies (ICT) in support of all the activities of business. Commerce constitutes the exchange of products and services between businesses, groups and individuals and can be seen as one of the essential activities of any business. Electronic commerce focuses on the use of ICT to enable the external activities and relationships of the business with individuals, groups and other businesses.

E-Commerce Is a particular form of e-Business. Electronic business methods enable companies to link their internal and external data processing systems more efficiently and flexibly, to work more closely with suppliers and partners, and to better satisfy the needs and expectations of their customers. Compared to e-Commerce, e-Business is a more generic term because it refers not only to information exchanges related to buying and selling but also servicing customers and collaborating with business partners, distributors and suppliers.

E-Business encompasses sophisticated business-to-business interactions and collaboration activities at a level of enterprise applications and business processes, enabling business partners to share in-depth business intelligence, which leads, in turn, to the management and optimization of inter-enterprise processes such as supply chain management. More specifically, e-Business enables companies to link their internal and external processes more efficiently and flexibly, work more closely with suppliers and better satisfy the needs and expectations of their customers.

In practice, e-business is more than just e-commerce. While e-business refers to more strategic focus with an emphasis on the functions that occur when using electronic capabilities, e-commerce is a subset of an overall e-business strategy. E-commerce seeks to add revenue streams using the World Wide Web or the Internet to build and enhance relationships with clients and partners and to improve efficiency using the Empty Vessel strategy. Often, e-commerce involves the application of knowledge management systems.

E-business involves business processes spanning the entire value chain: electronic purchasing and supply chain management, processing orders electronically, handling customer service, and cooperating with business partners. Special technical standards for e-business facilitate the exchange of data between companies. E-business software solutions allow the integration of intra and inter firm business processes. E-business can be conducted using the Web, the Internet, intranets, extranets, or some combination of these.

Basically, electronic commerce (EC) is the process of buying, transferring, or exchanging products, services, and/or information via computer networks, including the internet. EC can also be benefited from many perspective including business process, service, learning, collaborative, community. EC is often confused with e-business.

In e-commerce, information and communications technology (ICT) is used in inter-business or inter-organizational transactions (transactions between and among firms/organizations) and in business-to-consumer transactions (transactions between firms/organizations and individuals).
In e-business, on the other hand, ICT is used to enhance one's business. It includes any process that a business organization (either a for-profit, governmental or non-profit entity) conducts over a computer-mediated network.
A more comprehensive definition of e-business is: "*The transformation of an organization's processes to deliver additional customer value through the application of technologies, philosophies and computing paradigm of the new economy*."

Three primary processes are enhanced in e-business:

- **Production processes**, which include procurement, ordering and replenishment of stocks; processing of payments; electronic links with suppliers; and production control processes, among others;
- **Customer-focused processes,** which include promotional and marketing efforts, selling over the Internet, processing of customers' purchase orders and payments, and customer support, among others
- **Internal management processes**, which include employee services, training, internal information-sharing, video-conferencing, and recruiting. Electronic applications enhance information flow between production and sales forces to improve sales force productivity. Workgroup communications and electronic publishing of internal business information are likewise made more efficient.

E-Business goes far beyond e-commerce or buying and selling over the Internet, and deep into the processes and cultures of an enterprise. It is the powerful business environment that is created when you connect critical business systems directly to customers, employees, vendors, and business partners, using Intranets, Extranets, ecommerce technologies, collaborative applications, and the Web.

E-business is a more strategic focus with an emphasis on the functions that occur when using electronic capabilities while E-commerce is a subset of an overall e-business strategy. E-commerce seeks to add revenue streams using the World Wide Web or the Internet to build and enhance relationships with clients and partners and to improve efficiency while Electronic business methods enable companies to link their internal and external data processing systems more efficiently and flexibly, to work more closely with suppliers and partners, and to better satisfy the needs and expectations of their customers.

E-Business is at the enterprise application level and encompasses sophisticated b2b interaction and collaboration activities. Enterprise Application Systems such as ERP, CRM, SCM form an integral part of e-Business strategy and focus.

Critical Factors with respect of e-Business

E-Business supports business processes along the entire value chain: Electronic purchasing (E-Procurement), SCM (Supply Chain Management), Processing orders electronically, Customer Service & Co-operation with business partners.

One of the objectives of e-Business is to provide seamless connectivity and integration between business processes and applications external to an enterprise and the enterprise's back office applications sucha as billing, orger processing, accounting, inventory and receivables, and services focused to total supply chain management and partnership including product development, fulfillment, and distribution. In this respect, e-Business is much more than e-Commerce.

To succeed in e-Business it is crucial to combine technological developments with corporate strategy that redifines a company's role in the digital economy while taking into account its various stakeholders. It is imperative to understand the issues, evaluate the options, and develop technology orientation plans. An e-Business strategy helps organizations identify their e-Business concerns, assess their information needs, analyze to what degree existing systems serve these objectives, pinpoint specific improvements, determine the development stages of e-Business solutions and attain concrete and measurable results. Thus, it is clear that e-Business solutions are not only about technology.

A classic example is SAP systems integrations for any organization. This itself is taken up as a project and executed with great attention to detail. A minute logical error in interpretation of the firm's objectives could result in the entire system being re-worked from scratch.

E-Business allows for redefinition of value, competitiveness and the very nature of transactions and it affects all areas of an organization. It is crucial to combine technology and business strategy while taking into account various stakeholders

An E-business Strategy helps to

- Identify e-business concerns
- Assess info needs
- Analyze existing systems
- Improvements required in existing systems
- Determine the stages of development of solutions
- Attain concrete and measurable results.

Characteristics of e-Business

To emphasize, e-Business is not simply buying and selling but encompasses the exchange of many kinds of information, include online commercial transactions. E-Business is about integrating external company processes with an organization's internal business processes; as such, a variety of core business processes could exploit an e-Business infrastructure.

These include among others: -

- Collaborative Product Development
- Collaborative Planning, Forecasting and Replenishment
- Procurement and Order management
- Operations and Logistics

Collaborative Product Development

This is one of the fastest growing technologies in engineering with some form of solutions being implemented in a range of industries such as automotive, aerospace, agricultural machinery etc. It contributes towards making products in a short time span while maintaining quality and reducing cost.

It also aids in maximizing time-to-market benefits while maintaining control over product development information. By integrating design and testing cycles of products with those of suppliers, a firm can shorten the complete cycle of its products. This clearly, reduces the total cost of the product cycle, & even more importantly, it reduces the time that is needed to bring products to the marketplace. Collaborative product development solutions offer ERP integration and SCM.

Collaborative Planning, Forecasting and Replenishment

This is a process in which Manufacturers, Distributors and Retailers work together to plan, forecast and replenish products. In e-Business relationships collaboration takes the form of sharing information that impacts inventory levels and merchandise flow.

Collaboration points: sales forecasts, inventory requirements, manufacturing and logistic lead times, seasonal set schedules, new/remodel storage plans, promotional plans etc

Goal: To get the partners to work together to improve lower supply cycle times, improve customer service, lower inventory costs, improve inventory levels and achieve better control of planning activities

Procurement and Order management

Electronic procurement or E-Procurement can achieve significant savings and other benefits that impact the customer. To support procurement and order management processes, companies use an integrated electronic ordering process and other online resources to increase efficiency in purchasing operations.

Benefits: cost savings, better customer service by controlling the supply base, negotiating effective buying preferences, and streamlining the overall procurement process.

Operations & Logistics

Logistics is that part of the supply chain process that plans, implements and controls the efficient, effective flow and storage of goods, services and related information from the point of origin to point of consumption in order to meet customer requirements. To make this happen, transportation, distribution, warehousing, purchasing & order management functions must work together. Logistics in the e-Business era is all about Collaboration - the sharing of critical and timely data on the movement of goods as they flow from raw material, all the way to the end-user.

Operations and Logistics processes are based on open communication between networks of trading partners where integrated processes and technology are essential for high performance logistics operations. These solutions help manage the logistics process between buyers and suppliers, while eliminating costly discrepancies between purchase order, sales order and shipping information. By eradications these variances and inconsistencies improvements in the supply chain may result from the elimination of mixed shipments and shipment discrepancies, and the reduction of inventory carrying costs for the customer. At the same time this increases customer satisfaction through improved delivery reliability and improved efficiencies in receiving operations.

Furthermore, there are critical elements to e-business models as well. They are as follows:

- A **shared digital business infrastructure**, including digital production and distribution technologies (broadband/wireless networks, content creation technologies and information management systems), which will allow business participants to create and utilize network economies of scale and scope.
- A **sophisticated model for operations,** including integrated value chains-both supply chains and buy chains.
- An **e-business management model,** consisting of business teams and/or partnerships;
- **Policy, regulatory and social systems** - i.e., business policies consistent with e-commerce laws, tele-working/virtual work, distances learning, incentive schemes, among others.
- **Ease of Automated Processing** - A payer can now cheaply and easily automate the generation and processing of multiple payments with minimal effort. Previously, the dependency upon banks to handle most payments and the lack of a cheap, ubiquitous communications technology made automation of payment processes expensive and difficult to establish.
- **Immediacy of result** - Payment immediacy occurs because automation and the ability for the intermediate systems and providers to process payments in real-time. With the more manual, paper-based systems there was always a time delay due to the requirement for human intervention in the process.
- **Openness and accessibility -** The availability of cheap computing and communications technology and the appropriate software enables small enterprises and individuals to access or provide a range of payment services that were previously only available to large organizations via dedicated networks or the transactional processing units of banks.
- **Loss of collateral information** - The new technology dispenses with, or alters, *collateral information* accompanying transactions. This information has traditionally been part of the transaction, and has been relied upon by the transacting parties to validate individual payments.
- Collateral information can be defined as information:
- Which is not essential to the meaning and intent of a transaction;
- Which is typically incidental to the nature of the communications channel over which the transaction is conducted; but nevertheless provides useful contextual information for one or more of the parties to the transaction?
- Collateral information can include many things ranging from tone of voice in a telephone call to the business cards and letterheads and apparent authority of the person with whom you are dealing.

- **Globalization -** Globalization, or the minimization of geographical factors in making payments, has been an obvious aspect of the new payments systems. Its affect is upon areas such as size of the payments marketplace, uncertainty as to legal jurisdiction in the event of disputes, location and availability of transaction trails, and the ability of a payment scheme to rapidly adapt to regulatory regimes imposed by one country by moving to another.
- **New business models** - New business models are being developed to exploit the new payment technologies, in particular to address or take advantage of the disintermediation of customers from traditional payment providers such as banks. In this context, disintermediation is where the technology enables a third party to intervene between the customer and the banking system, effectively transferring the customer's trusted relationship with the bank to the new party.

Elements of an e-Business solution

The vision of e-Business is that enterprises will have access to broad range of trading partners to interact and collaborate with and not only buy and sell more efficiently. Also, it is expected that e-Business will contribute towards the agility of business organizations and with that to reaching higher levels of customization. In this way, an organization can maximize supply chain efficiency, improve customer service and increase profit margins. Hence, the need to make mission critical processes:

Inventory, Accounting, Manufacturing and Customer Support: These, must be able to interact with each other by becoming web-enabled. This is achieved by ERP, CRM and other systems by making use of distributed applications that extract data and launch business processes across many or all of the above processes.

The key elements of an e-Business solution are:

1. Customer Resource management(CRM)
2. Enterprise resource planning (ERP)
3. Supply Chain Management (SCM)
4. Knowledge Management
5. e-Markets

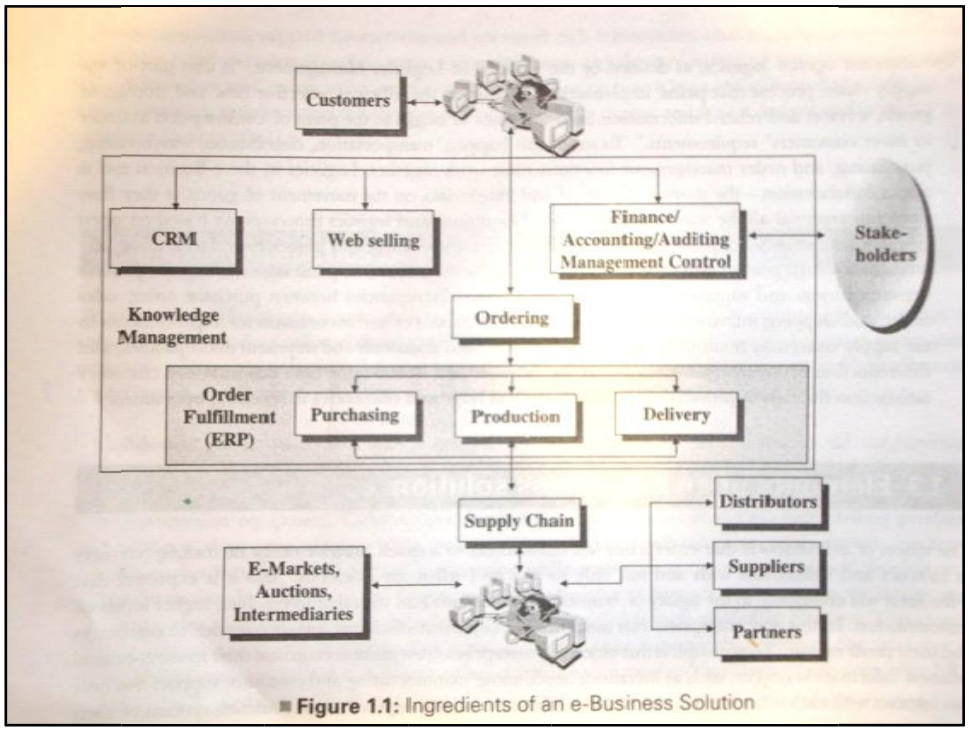


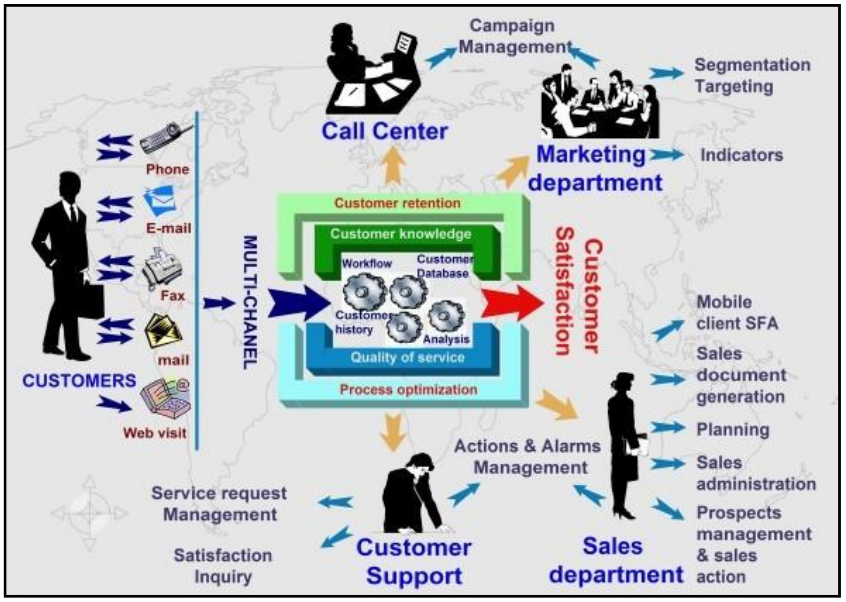
Figure 1.1: Ingredients of an e-Business Solution

Customer relationship management (CRM)

CRM systems are “front-office” systems which help the enterprise deal directly with its customers. CRM (definition) is the process of creating relationships with customers through reliable service automated processes, personal information gathering, processing and self-service through the enterprise in order to create value for customers.

There are 3 categories of user applications under CRMs:

- **Customer-facing applications:** Applications which enable customers to order products and services
- **Sales-force facing applications:** Applications that automate some of the sales and sales-force management functions, and support dispatch and logistic functions.
- **Management-facing applications:** Applications which gather data from previous apps and provide management reports and compute Return on relationships(RoR) as per company’s business model

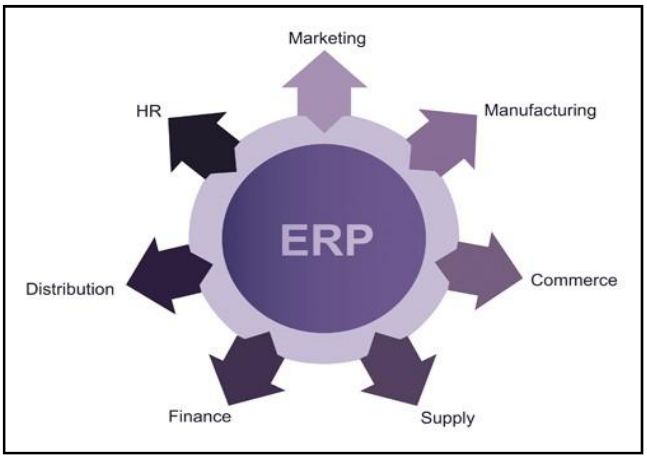


Enterprise Resource Planning (ERP)

ERPs are often called “back-office” systems. ERP systems are management information systems that integrate and automate many of the business practices associated with operations or production aspects of a company. ERP software can aid in control of many business activities such as sales, delivery, production, billing, production, inventory, shipping, invoicing and accounting.

A typical ERP system is designed around these 4 primary business procedures: -

- **Production:** manufacturing, resource planning and execution process
- **Buying a product:** procurement process
- **Sales of a product and services:** customer order management process
- **Costing, paying bills, and collecting:** financial/management accounting and reporting process.

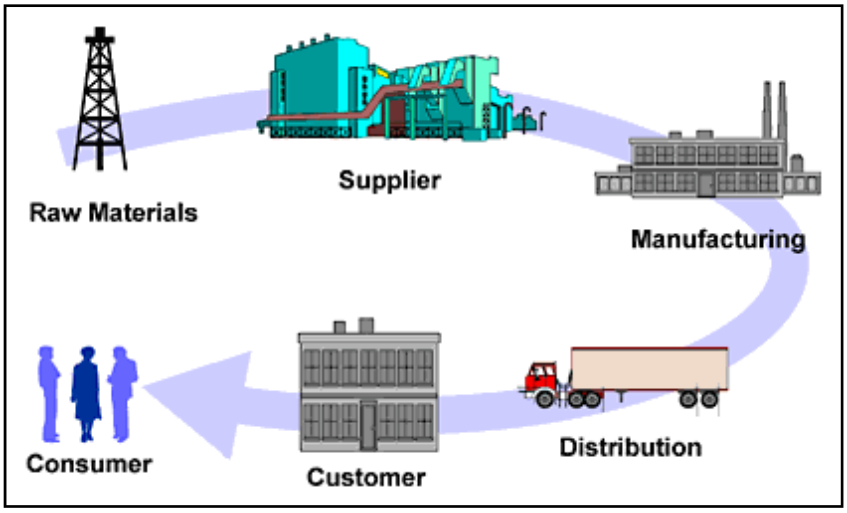


Supply Chain Management (SCM)

Supply chain (definition) is a network of facilities and distribution options that perform the functions of procurement of materials, transformation of these materials into intermediate and finished products, and distribution of these finished products to customers. SCM deals with the planning and execution issues involved in managing a supply chain.

Supply chain has 3 main parts

- **Supply side:** concentrates on how, where from, and when raw materials are procured and supplied to manufacturing.
- **Manufacturing side:** converts raw materials to finished products.
- **Distribution side:** ensures that finished products reach the final customers through a network of distributors, warehouses and retailers.



Knowledge Management

This relates to the identification and analysis of available and required knowledge assets and related processes. Knowledge assets encompass two things Information and Experience. Knowledge assets comprise of all knowledge that a business has or needs to have in order to generate profits and add value.

Knowledge management includes the subsequent planning and control of actions to develop both the knowledge assets and the processes to fulfill organizational objectives. Knowledge is a strong denominator of a business model and determines business competencies especially when unique to the business and so must be kept in-house.

E-Markets

E-Market is an electronic meeting place for multiple buyers and sellers providing many participants with a unified view of sets of goods and services, enabling them to transact using many different mechanisms. An e-Market uses Internet technology to connect multiple buyers and suppliers.

E-Business Roles and their challenges

- There are two main roles in the E-business scenario:
  - The Buyer: Buyers are organizations that purchase goods and services directly from Suppliers.
  - The Supplier: Suppliers are organizations that market and sell goods or services directly to buyers or indirectly through diverse sales channels including Web-based procurement systems and electronic marketplaces.
- Suppliers typically provide buyers with web-based services necessary for completing e-Business transactions.
- Buyers (customers) can thus review product information, receive customer service, ordering services and customization support facilities an can submit or modify orders.
- An additional role is that of **Market Makers** that are third party organizations that run e-markets.
- Each role has distinct business and technical challenges, but they all coalesce around a common point.
- For buyers as well as for suppliers, the primary challenge is the ability to reach a critical mass of trading partners and transaction volume to sustain their business.
- For suppliers especially, the following challenges exist:
  - Managing multiple selling channels, based on various technologies, protocols, data formats, and standard business processes.
  - Having the ability to take multiple types of orders once the customer has decided to conduct e-Business –enabled order management through the various selling channels.
  - Having the ability to differentiate and customize products and services from other suppliers, and offering them through the various selling channels.
  - Having the ability to adapt and grow the e-Business without incurring drastic technology changes, organizational restructuring.

- And sweeping changes in the business process, or radical new investments.
- To meet the needs of buyers and suppliers, e-Business strategy and solutions must be built on the following basic principles:
  - Empowering suppliers & buyers:
    - Different channels.
  - Enabling suppliers of all sizes:

E-Business Requirements

- **Identify/measure quantifiable business objectives:** companies must accurately measure the impact an e-Business initiative has on their business processes and decide whether this initiative is worth pursuing and has sustainable long-term effects
- **Ensure organizational/operational flexibility:** Enterprises must reposition themselves in their mission, structure and execution to prosper in a substantially more dynamic environment.
- **Rethink entire company supply chains:** companies must rethink their entire supply chains in order to optimize performance and value as they seek to better integrate with suppliers and customers, share information, inter-link processes, and outsource manufacturing logistics systems and maintenance activities.
- **Transform the company to a process-centric form:** Companies must be conceptualized as a set of business processes with more emphasis on maximizing the efficiency of processes rather than departmental or functional units.
- **Define Business processes:** companies must create models of existing processes and interactions determining the relevant events, time frames, resources and costs associated with business processes, hence making them well-defined and measurable
- **Understand Security requirements:** the breadth of access and interaction requirements of a e-Business solution requires the ability to provide controlled and focused access by all the users.
- **Align business organizations with a flexible IT architecture:** in response to demands for end to end e-Business solutions, companies are expanding their applications to include enhanced integration capabilities. This includes integration of business processes at varied levels from applications and data across(and within) organizations.
- **Establish ubiquity within standards:** None of the many integration technologies available from various IT vendors has achieved complete coverage. These do work within organizations but not across global enterprises and between separate enterprises. Attempts are made to establish open standards for interoperability.
- A number of business and tech. driven requirements are compelling forces that enable successful development & deployment of integrated end-to-end e-Business applications. Some of these are:
  - Efficient business process management technology
  - Efficient b2b communication
  - Efficient enterprise application integration technology
  -

Other categorizations view the problem differently.

A more basic approach to viewing e-Business requirements is as follows: -

- **Trust** - The biggest requirement for running a successful e-business is trust. In this age of Facebook and MySpace, online merchants may think that privacy of a customer's information isn't important, but the opposite is true.
  - Thus, businesses must be trustworthy to operate online. Consumers will not simply give their financial information to just anyone, so a site will lose business if consumers do not feel comfortable that it is a reliable, upstanding company.

- Companies must have comprehensive privacy policies and stick with them. Another good idea is to get digital certificates and TRUSTe seals, which are awarded by third-party organizations after they research the legitimacy of an online website.
- Such awards put consumers' minds at ease. Finally, even if an e-business does all this, it must also be trustworthy in the sense of fulfilling its promises: be up front with consumers about pricing and delivery times.

- **Privacy policy** - In addition to the way privacy laws apply in the "real" world, there are some special things to think about when dealing with the Internet and e-business.
  - You should fully understand how your website fits into privacy law requirements.
  - If your website collects personal information, you should develop a proper and legally compliant privacy policy and post it in a readily visible location on your website.
  - If you use cookies or similar means to track visitors, depending on how you do that, you may still need to develop and post a policy.
  - Online profiling may require the consent of the individual depending on the circumstances.
  - Keep in mind that people do look for privacy policies so, without a policy, you may lose prospective customers.
  - A properly drafted privacy policy or statement will not only minimize your legal exposure, it can serve a marketing function as well, allowing you to attract and retain customers who otherwise might not be as inclined to deal with you.
  - Do not create a policy and then fail to follow it precisely. This is an invitation for disaster, including not only possible legal problems, but also injury to your reputation and goodwill.
- It is important to not just let the policy sit once it has been posted. It should be revisited regularly to determine whether or not it is still accurate and to evaluate whether or not it should be revised to assist you in your business goals and objectives.

- **Strategy -** E-commerce merchants must also have a strategy to succeed in the online marketplace. Many people start websites because they think it is a quick and easy way to make cash, but in fact it takes a much greater investment than most people expect.

- Therefore, before launching a site, businesses must have strategies to handle issues large and small:
  - How consumers will place orders,
  - How deliveries will be made,
  - How customer service issues will be handled?
  - More broadly, how much do owners expect to earn over a certain period, how will consumers find the site, and how will success be judged.
  - Online merchants without strategies will soon be overwhelmed by such issues.

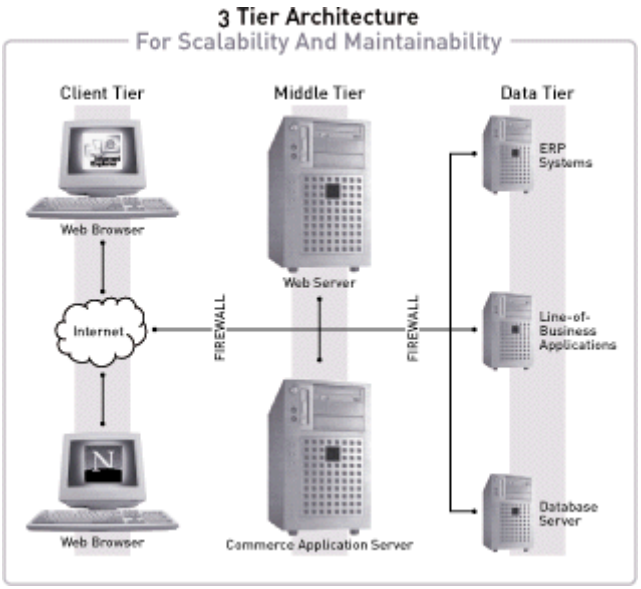
- **Suitability -** Finally, merchants must decide if their products are suitable for the web. Requirements for successful e-businesses concern the goods and services themselves:
  - Can they be delivered quickly and cheaply?
  - Do they appeal to people outside a small geographic area?
  - Will going online save money?
  - Will the benefits outweigh the costs?

Content Management	The ability for business managers to publish, reuse, and update information to meet changing business conditions
Personalization	The ability to target, format, and automatically adapt content for specific users to meet their preferences and your business requirements
Security	The ability to give users access to information in core business systems and ensure that they see only what they are authorized to see
Integration	The ability to present existing business applications and data stores as a unified information stream for the benefit of customers, partners, and employees
System Manageability	The ability for business managers and partners to administer and manage user accounts and other aspects of the eBusiness system in response to changing business relationships
Time-To-Market	The ability to mount eCommerce initiatives in a timely and cost-effective way, so that they enhance rather than detract from the bottom line

- **Technological Requirements:**
  - **Achieving Real - Time Flexibility:** In theory, digital things are easier to change than physical things. It is faster to edit a memo using a word processor than a typewriter (and you don't get ink on your fingers).
  - But when programming is required to change content or access policies, maintaining a complex Web site can range from onerous to impossible. Market factors change in real time, and so must the logic and content of an e-Business site.
  - To achieve this vision, the next generation of e-Business systems must provide a framework for automated information exchange between all the stakeholders in a business.
  - These new frameworks are designed for flexibility so companies can change content and business logic in real-time to meet changing business needs and market conditions.
  - This adaptability comes from a set of core services, common to all applications, which enable rapid deployment of new applications and new information and which work together to create a compelling, unified e-Business environment.
  - An e-Business framework must include packaged, ready-to-deploy services for:
  - **An Architecture For e-Business** - As e-Business moves beyond simple transactions to encompass all the complex processes through which a company provides value, information systems must orchestrate the function of enterprise applications and information resources for total information flow. And they must empower business people with the tools to manage content publishing, delivery, and access, so that business results don't depend on the IT department's programming backlog.
  - **Three - Tier Object - Centered Design** - To achieve true, real-time e-Commerce, next-generation e-Business systems must be built around a 3-tier application paradigm with a clear abstraction and true separation of user interface presentation, business logic, and content. Separation and abstraction of these layers is achieved through the use of business objects, particularly in the middle layer.
  - When separating presentation, application, and Data Logic three things must be considered:
    - **User Interface** - The user interface must support a variety of interface mechanisms, including Web browsers for users, business managers, designers and desktop applications for developers.
    - **Business Logic** - The middle tier must not only implement and execute business logic, it must also provide the framework of services that enable e-Business, including security services,

transaction services, and caching, pooling, and other load balancing services to improve overall system performance.

- **Content** - The content layer includes corporate databases, document stores and other knowledge repositories



- **All Objects Are Not Created Equal** - The overall architecture of an e-Business system is important, but proper abstractions achieved through object technology are the foundation of a flexible e-Business system. Correct separation of presentation, business logic, security functions, and content determines the flexibility of the system and the pace and effectiveness of e-Business processes. To deliver truly dynamic, real-time communication, these relationships must be established on a per-transaction basis, as each page is assembled for delivery to a user e-Business processes lend themselves to this kind of abstraction.
- **Bringing Order To Content Management -** As companies move more of their business processes onto the Web in search of greater sales or efficiency, Web sites are growing in size and complexity. Static Web sites often consist of hundreds or even thousands of Web pages, and tens of thousands of lines of code. Multi-media sites are becoming the standard, with everything from sophisticated graphics and animation to audio and video. Enterprise Web sites must integrate multiple applications from the back-office to the supply and sales chain, while maintaining security and the integrity of business information. As sites become larger and more complex, traditional Web publishing systems, with their hard-coded Web page content, become unmanageable. Content creators swamp programmers with requests for new Web pages, the approval process bogs down, and users no longer have access to current content.

Business Objects	Such as Person, Document, or Product objects are the content base for the Web site and the basis of a dynamic, scalable security scheme. Access to each entity can be controlled individually, based on group memberships.
Queryable Groups of Business Objects	Queries are used to create interesting groupings such as all partners who have sold the XYZ product in the last year, or all customers who downloaded the ABC white paper from the Web site this month. These can be used to target and limit information to specific users, and to define "roles" which can be used to delegate privileges such as content publishing or system administration.
Graphical Layouts	Determine the arrangement of graphical elements on Web pages, but which should not determine the content which will be displayed in those graphic elements.

- **Dynamic Web Environment** - The graphical layouts used in next-generation e-Business systems are more "intelligent" and manageable than the templates used in traditional Web publishing. While both control placement of graphic elements, style, etc., templates access content through business logic hard-coded into the body of the page. Pages with different content, however similar, require different source files. **Layouts**, on the other hand, are a next-generation approach that does not embed business logic in the presentation objects. A layout controls only style and placement of elements on the page. The logic that determines content is separate from the layout, and can be changed and maintained independently.
- **Content Management Tools -** Content Management tools must enable content to grow and change at Web speed.
- **Team Content Development -** Publishing content to the Web and extending the functionality of a site takes a whole team: developers to build site structure and implement business rules, designers to create page layouts and define a consistent look and feel, and business managers to define business rules and contribute content.
- **Collaboration Across The Extended Enterprise -** Publishing and managing Web content typically involves an approval process and some administrative work.
- **Centralized Rules - Based Content Management:** Anyone should be able to manage site content simply by defining a few document characteristics when a document is published. With characteristics such as a document's type (for example, "data sheet"), format, and activation/expiration dates in place, links to the document can be automatically populated throughout the site, and document visibility and "document migration" can be automated.
- **Customized Content Delivery**
- **Pervasive Personalization**
- **Knowledge-Based Personalization -** Effective personalization depends on the ability to customize a user's experience based on a rich, centrally stored user profile: in essence, a knowledge base that consists of user information and expertise on how to apply that information. This kind of knowledge base cannot be bolted onto a brochure ware Web site. The ability to gather and apply user-related knowledge must be integrated into the e-Business system from day one, so that information can be contributed, shared, and leveraged by all the applications in the system.
- **User profiles** are the crown jewels of an e-Business strategy. The quality of profiles determines the degree to which the user experience can be personalized. Profiles can and should be built through both explicit and implicit mechanisms.
- **Inclusive Security**
- **Scalability To Compete -** As more and more processes are adapted to e-Business, a Web site may grow to support thousands of users, millions of documents and millions of transactions each day. An e-Business system must have the power to perform fast and reliably, as a business grows, while delivering the dynamic, personalized content necessary to achieve business goals.
- **Enterprise Integration and Transaction Monitors -** Any business process can be "Web-ified" with a CGI interface or a few server pages. But isolated, e-Business is about providing new value by doing business in a fundamentally new way. Integration is the goal and the heart of e-Business: integrating and exposing applications and content in a personalized way to speed, scale and improve business processes and to engage, involve, and build lasting relationships with customers and business partners.
- **Transaction management** guarantees that users have a consistent view of business information. For example, the e-Business system should prevent the customer from completing an order based on one price, then being charged based on the new price. Transaction management also prevents inaccurate results based on system failures (e.g., the system goes down and loses an order but continues to process the billing using already-transmitted credit card information). Robust system logs can help coordinate updates across multiple data sources from multiple vendors or roll back changes in case of system failure.

- **Delegated System Management -** e-Business systems are distributed by their very nature, coordinating information sharing among applications, business functions and departments, and partners up and down the supply chain. Bringing business processes to the Web increases the complexity of the e-Business site, and growing and changing numbers of users and applications increase the complexity of managing a site. No centralized IT department could effectively maintain current accounts or access privileges for all users, inside and outside the company. Most Web sites today are not sophisticated enough to reach this roadblock, but as businesses open and extend their processes via the Web, system manageability will become an increasingly serious issue.
- **Time to Market –**Time to market must be minimal as delays may result in losing the benefit of e-Business integration.

Impacts of e-Business

- **Improved operational efficiency and productivity:** by eliminating operational waste and automation of inefficient business practices, organizations can realize productivity gains
- **Reduction in operating costs and costs of goods and services:** by connecting directly with suppliers and distributors, organizations can realize more efficient processes that result in reduced units of cost for products or services and lower prices to customers while achieving economies of scale.
- **Improved competitive position:** global reach, rapid growth, efficient reduction of product time to market and optimization of product distribution channels all contribute to superior competitive position.
- **Penetration into new markets through new channels:** with e-Business location is of no consequence when it comes to reaching customers.
- **Improved communication, information and knowledge sharing:** alignment of key supply chain partners with an organization's internal strategies helps exploit their expertise and knowledge, hence creating opportunity to secure long-term business by embedding their process and procedures in those of their customers' supply chains.
- **Harmonization and standardization of process**
- **Improved internal information access**
- **Improved relationships with suppliers and improved customer service**

Inhibitors of e-Business

- **Management/Strategy issues**
  - e-business strategy
  - Organizational changes required by e-business
  - Management attitudes and organizational inflexibility
- **Cost/financing issues**
  - Costs of implementation
  - Calculating the Return on Investment (ROI)
- **Security and Trust Issues**
- **Legal Issues**
  - Few companies are familiar with the rules and regulations that apply to an online environment.
    - This leads to Uncertainty.
  - Different strokes for different folks!
- **Technological Concerns**
  - Integration Issues
- **Arguments against Investment**
  - Uncertainty & Fear

E-Business Strategies

What is an E-Business Strategy?

- E-Business has triggered new business models, strategies and tactics that are made possible by the internet and other related technologies.
- In order to compete in the marketplace, it is essential for organizations to establish strategies for the development of an e-business.
- E-Business strategy can be viewed via two different viewpoints, which are explained below.
- One view defines strategy as plans and objectives adopted to achieve higher-level goals.
- In that sense, a strategy is developed to achieve a goal like implementing organizational change, or a large software package such as an ERP-system.
- Strategy may also relate to plans concerning the long-term position of the firm in its business environment to achieve its organizational goals.
- Based on the above, we arrive at a common definition for an e-Business Strategy.
- An e-Business strategy is the set of plans and objectives by which applications of internal and external electronically mediated communication contribute to the corporate strategy.

- Strategic planning comprises a distinct class of decisions (a plan is a set of decisions made for the future) and objectives, and has to be positioned next to tactical planning (structuring the resources of the firm) and operational planning (maximizing the profitability of the current operations).
- Strategy is concerned with changes in the competitive environment that may trigger strategic changes for the individual firm and so affect its roles and functions in the market.
- Reassessment of strategy may occur due to:
    - New Products
    - Changing customer preferences
        - Flowers: Roses / Carnations -> Orchids
        - A few years back when people went to the florist, they generally picked up Roses or Carnations etc. Now, they prefer Orchids. This is an example of changing customer preferences. A global notion is that a customer does not realize the utility of feel the need for a product until it is offered to him / her.
    - Changing demand patterns
    - New competitors
- The frequency, dynamics and predictability of the above changes dictate the intensity of the strategic planning activity of the firm.

- So, e-Business strategy (revised) is:
    - The set of plans and objectives by which applications of internal and external electronically mediated communication contribute to the corporate strategy.
- E-Business strategy may be implemented for:
    - Tactical purposes: Mail -> EDI ->XML-FDI
    - Achieving corporate strategy objectives
- E-Business is strategic in nature.
    - The idea is to create a preferably sustainable & competitive position for the company.
        - This is achieved by integration of the Internet and related technologies in its primary processes.
- E-Business must not only support corporate strategy objectives but also functional strategies (SCM, Marketing)

- **Supply Chain Management Strategy**

- Based on value chain analysis for decomposing an organization into its individual activities and determining value added at each stage.
- Gauge efficiency in use of resources at each stage.

- **Marketing Strategy**
  - Is a concerned pattern of actions taken in the market environment to create value for the firm by improving its economic performance.
  - Focused on capturing market share or improving profitability via brand-building etc.
  - Operates on CURRENT AS WELL AS FUTURE projections of customer demand.
- **Information Systems Strategy**
  - How to leverage information systems in an organization to support the objectives of an organization in the long run.

- **E-Business strategy is based on corporate objectives.**

Strategic Positioning

Strategic positioning means that a firm is doing things differently from its competitors in a way that delivers a unique value to its customers. There are 6 fundamental principles a firm must follow to establish and maintain a distinctive strategic position:

1. Start with the right goal: superior long term ROI.
2. Strategy must enable it to deliver a value proposition different from competitors.
3. Strategy must be reflected in a distinctive value chain.
4. Accept tradeoffs for a robust strategy.
5. Strategy must define how all elements of what a firm does fit together.
6. Strategy must involve continuity of direction.

Levels of e-Business Strategies

Strategies will exist at different levels of an organization. Strategic levels of management are concerned with integrating and coordinating the activities of an organization so that the behavior is optimized and its overall direction is consistent with its mission. Ultimately e-Business is about communication, within business units and between units of the enterprise as well as organizations.

1) **Supply Chain or Industry Value Chain level**
   - E-Business requires a view of the role, added value, and position of the firm in the supply chain.
   - Important issues that need to be addressed at this level are:
     i. Who are the firm's direct customers?
     ii. What is the firm's value proposal to the customers?
     iii. Who are the suppliers?
     iv. How does the firm add value to the suppliers?
     v. What is the current performance of the Supply Chain in terms of revenue and profitability, inventory levels etc?
     vi. More importantly, what are the required performance levels?
     vii. What are the current problems in the chain?
   - This sort of analysis give insight into in upstream (supplier side) and downstream (customer side) data and information flows.
2) **The Line of Business or (Strategic) Business Unit level**
   - Understanding the position in the value chain is a starting point for further analysis of how Internet-related technologies could contribute to the competitive strategy of a business.
   - This is the level where competitive strategy in a particular market for a particular product is developed (Strategic Positioning).

- There are four generic strategies for achieving a profitable business:
    i. **Differentiation:** This strategy refers to all the ways producers can make their product unique and distinguish them from those of their competitors.
    ii. **Cost:** Adopting a strategy for cost competition means that the company primarily competes with low cost; customers are interested in buying a product as inexpensively as possible. Success in such a market implies that the company has discovered a unique business model which makes it possible to deliver the product or service at the lowest possible cost.
    iii. **Scope:** A scope strategy is a strategy to compete in markets worldwide, rather than merely in local or regional markets.
    iv. **Focus:** A focus strategy is a strategy to compete within a narrow market segment or product segment.

3) The Corporate or Enterprise level

- This level comprises a collection of (strategic) business units.
- This level addresses the problem of synergy through a firm-wide, available IT infrastructure.
- Common e-Business applications throughout the organization are needed for two basic reasons.
- From efficiency point of view, having different applications for the same functionality in different areas of business is needlessly costly.
- From an effectiveness point of view, there is the need for cross Line of Business communication and share-ability of data.
- The emphasis in the business plans is on the customer, not the final product.
- These all become subjects of an enterprise-wide e-Business policy.

The changing competitive Agenda: Business & Technology Drivers

Business Drivers:

- Shift in economies from supply driven to demand driven
    - Causes a shift in intent of service and quality programs, the impetus for product development & the structure of the organization itself
    - One to One marketing
    - Mass Customization

Technological Drivers:

- Internet
    - Pervasiveness
    - Interactive Nature
    - Virtual Nature

Strategic Planning Process

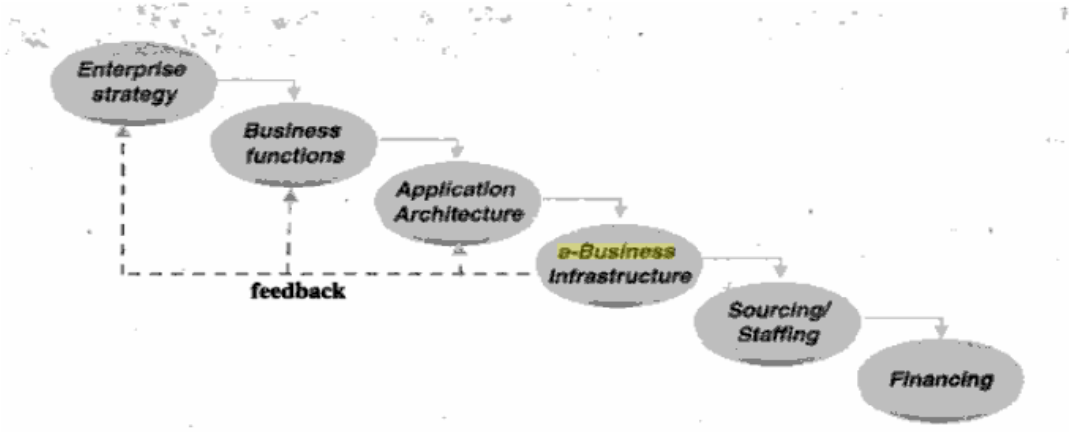
- The strategic planning process has the following steps:
- The strategic planning process starts with the establishment of the organization's mission statement.
    - The mission statement is a basic description of detailing the fundamental purpose of the organizations existence and encompasses strategy development, including determination of the organization's vision and objectives.
    - It is developed at the highest level of the organizations management, and provides a general sense of direction for all decision making within the firm.
- Strategic Analysis
    - This involves situation analysis, internal resource assessment, and evaluation of stakeholder's expectation.

- It will include
  - Environmental Scanning
  - Industry or market research
  - Competitor Analysis
  - Analysis of Marketplace Structure
  - Relationships with trading partners and suppliers
  - Customer Marketing Research
- Information is derived from the analysis of both internal and external factors.
- Internal Factors:
  - Human resources
  - Material resources
  - Informational resources
  - Financial resources
  - Structure
  - Operational Style
  - Culture
- External Factors:
  - Socio-cultural forces
  - Technological forces
  - Legal and regulatory forces
  - Political forces
  - Economic forces
  - Competitive forces
- Any realistic new plan will have to reflect the reality of both the external world and the internal dynamics of the corporation.

- Strategic Choice
  - It is based on the strategic analysis and consists of four parts:
    - Generation of strategic options
    - Highlighting possible courses of action
    - Evaluation of strategic options on their relative merits
    - Selection of strategy
  - Strategic choice results in Strategic Planning, which is concerned with the organizing and detailing of all the strategies that will be undertaken throughout the organization.
  - Planning includes strategy specification and resource allocation.
  - It commences with corporate-level planning that determines the overall direction for the organization.
  - This drives Division (or Strategic Business Unit) level planning which deals with groups of related products offered by the organization.
  - These plans in turn become the starting point for operating (or functional) level planning, which involves more local plans within specific departments of the organization.
- Implementation
  - This relates to the actual tasks that must be executed in order to realize a plan and translates strategy into action.
  - It includes monitoring, adjustment, control as well as feedback.

Strategic Alignment

- In the 1980s the concept of alignment between business and IT was developed.
- According to this concept it is not only feasible to design and build a technically sophisticated infrastructure for e-Business, but also to formulate business strategies that complement and support this infrastructure.
- One of the major issues regarding an enterprises investment in IT is whether this is in harmony with its strategic objectives.
- This state of harmony is referred to as alignment.
- Alignment is complex, multifaceted and almost never completely achieved. It is about continuing to move in the right direction and better aligned than the competitors.
- Any e-Business strategy should articulate an enterprise's intention to use information technology based on business requirements.
- When formulating the IT strategy, the enterprise must consider:
  - Business objectives and the competitive environment
  - Current and future technologies and the costs, risks, and benefits they can bring to the business.
  - The capability of the IT organization and technology to deliver current and future levels of service to the business.
  - Cost of current IT, and whether this provides sufficient value to the business.
  - Lessons learned from past failures and successes.



Consequences of e-Business

- As e-Business is an information technology-enabled organizational phenomenon with economic consequences, economic theories appear to be particularly useful for analyzing the business effects.
- Strategy is about finding the right (external) fit between organization and environment. Different schools of thought have approached this problem from different angles.
- When analyzing the business effects of an e-Business, we will consider the following approaches:
  - The Theory of Competitive Strategy
  - The resource-base view
  - The theory of transaction costs
- **Theory of Competitive Strategy**
  - The structural attractiveness of a firm is determined by five underlying forces of competition:
    - The bargaining power of the customers
    - The bargaining power of the suppliers
    - The barriers to entry for new competitors
    - The threat of new substitute products or services
    - The competition among existing firms in the industry

- In combination, these forces determine how the economic value created by any product, service technology or way of competing is divided between companies in an industry.
- The bargaining power of customers for a firm could, for instance, depend on the degree of product differentiation, and the size of demand and supply. Switching costs are also very important: they answer the question of how much will it cost the customer to change to another supplier.
- The bargaining power of suppliers is dependent on a variety of factors, such as relative size, number of suppliers, that can deliver a critical resource, and so on. The Internet causes another specific threat from the perspective of IT suppliers; they may bypass their customers and directly approach the end-user.
- The barriers to entry for new competitors depend on how difficult it is to join the industry. Economic and technological thresholds may prevent outside potential competitors to come in . Economies of scale, necessary capital, and specialized expertise are important factors in this respect.
- The threat of substitute products depends on the question of whether other products can deliver added value for consumers instead of current products in the absence of switching costs. e.g. – The Internet is a serious threat to the Post Office.
- The level of competition among existing firms in the industry will depend on various factors like type of market, existing competitive behavior, and so on.

- **The Resource-Based View**
  - According to this theory of economic development, innovation is the source of value creation.
  - Several sources of innovation (hence, value creation) are identified:
    - The introduction of new goods or new production methods,
    - The creation of new markets,
    - The discovery of new supply sources,
    - And the reorganization of industries.
  - The resource-based view (RBV), which builds on the theory of economic development's perspective on value creation, regards a firm as a collection of resources and capabilities.
  - The RBV looks at available resources first to see how a position in the business environment can be acquired with them.
  - According to this view, a firm can build a strategic position by picking the right resources and building competencies that are unique and difficult to imitate.
  - Resources are considered the raw material for building competencies.
  - The RBV states that marshalling and uniquely combing a set of complementary and specialized resources and capabilities may lead to value creation.
  - A firm's resources and competencies are valuable if, and only if, they reduce a firm's costs or improve its revenues.
  - Core competencies of an organization encompass knowledge bases, skill sets, and service activities that can create a continuing competitive advantage.

- **Transaction Cost Economics**
  - Transaction Cost Economics attempt to explain firms' choices between internalizing and buying goods and services from the market.
  - According to transaction cost theory, exchanges with external firms entail a variety of co-ordination costs associated with various aspects of inter-firm transactions.
  - The central question addressed by transaction cost economics is why firms internalize transactions that might otherwise be conducted in markets. Thus, two key issues concerning firms are:
    - Which activities should a firm manage within its boundaries, and which activities should it outsource?
    - In which way should a firm manage its relationship with its customers, suppliers and other business partners?

- According to transaction cost economics, a firm has two options for organizing its economic activities: an internal hierarchical structure where it integrates the activity into its management structure, or a market-like relationship with external firms.
- Critical dimensions of transactions influencing the choice of the most effective governance form are:
  - Uncertainty
  - Exchange Frequency
  - Specificity of Assets enabling the exchange
- Transaction costs include the costs of planning, adapting, executing and monitoring task completion.
- Transaction cost Theory assumes that markets are not perfect, so lead to costs, like search and monitoring costs.
- As internet technology is expected to significantly reduce transaction costs, this theory provides a basis for assessing the effects of the Internet on new and existing business models.

Success factors for Implementation of e-Business Strategies

- Transforming an enterprise from a traditional organization to an e-business based organization's a complex endeavor.
- It is essential that senior management develops and endorses a broad strategic vision.
- Once the strategy has been determined and approved the implementation strategy has to be chosen.
- Two approaches prevail:
  - The top-down approach: According to this, business transformation is a business—wide phenomenon that can only be implemented business wide.
  - The bottom-up approach: In this approach, business re-engineering starts as an experiment in an inconspicuous part of an organization. Lessons are learnt from this experiment, and the knowledge is transferred to other parts of the organization.
- Although the bottom-up approach has strong support , especially in the case of innovation, central co-ordination of the transformation activity is mandatory.
- To provide for the central coordination, program management has to be instituted. A core part of program management is multi-project management , the main objectives of which are:
  - Recognize dependencies between projects
  - Share scarce resources in an overall efficient way
  - Systematically utilize experiences from single projects
- Program management is characterized by:
  - Program organization,
  - Policies,
  - Plans,
  - Communication,
  - Alignment.
- Leading a change project or business-wide initiative requires persons that plan the change and build business-wide support; these are called 'change agents'.
- Change Agents are part of the program management organization.
- In principle, everyone involved in a change project can assume the role of a change agent.
- Three types of change agent roles have been identified:
  - **Traditional:** In the traditional model, the Information System (IS) specialists focus on the delivery of the implementation of the technology, without considering the organizational aspects. Consequently they become technicians with a narrow area of expertise.
  - **Facilitator:** In the facilitator model, the central model is that people, not technologies create change. The change agent brings together all the conditions necessary for the change. In this model, the change agent remains 'neutral', the organization is responsible for the change.

- **Advocate:** In this role, change agents focus on inspiring people to adopt the change. Unlike the facilitator, he does not remain neutral, but uses any tactic (persuasion, manipulation, power etc) to make the changes accepted.

- Especially in the case of e-Business transformation, where organizational and IT changes relate to infrastructure and issues of commonality and interoperability, the advocate model seems to be appropriate.

Pressures Forcing Business Changes

- Competition
  - Fiercer & More Global
- Customers have become increasingly demanding
- Integrated Demand (Travel, Car-Rental etc.)
- Firms ask themselves "which of my competences are unique and of core importance?"
  - E.g. -> Bajaj exits the scooter market.
- Company configuration changes due to outsourcing and in-sourcing.

Business Models

There are various definitions for Business Models. The definitions change based on the paradigm and the context being applied. Let's look at each definition: -

- **Participants in a joint business venture:**
  - Specify the relationships between different participants in a commercial venture, the benefits & costs to each and the flows of revenues. It addresses a simple equation (profit= revenue-cost) irrespective of the model.
  - Describes how the enterprise produces, delivers and sells its products or services, thus showing how it delivers value to the customer and how it created wealth.
- **Process & structure of a business organization:**
  - Refers to the structures & processes in place to operationalize the strategy of business:
  - Can be described as:
    - An architecture for the product, service & information flows;
    - A description of the various business actors & their roles;
    - A description of the potential business benefits for the various actors;
    - A description of the sources of revenues.
- **Perspective of a marketplace.**
  - Definition can be analyzed from various perspectives:
    - B2B, B2C activities or both?
    - Position in the value chain?
    - Value proposition & target customers?
    - Specific revenue models for generation of it s income streams?
    - Representation? Physical or Virtual or combination?
- **Perspective of e-Business**
  - A descriptive representation of the planned activities of an enterprise that involves 3 integral components which specify: -
    - Internal aspects of a business venture
    - Type of relationships of the enterprise with its external business environment and its effective knowledge regarding these relationships
    - How the information assets are embedded in the business venture.
- **A business model can be viewed as an externalization of a firms internal business processes**
  - Does not involve internal business process complexity.
- **When taking the internal aspects of a business into account the following elements need to be defined:**

- Products or Services
- Sources of revenue
- Activities
- Organization of the firm

E-Business Models

E-Business models are classified as follows:

- **Internet Enabled**
  - Categorized based on increasing functionality, innovation, integration and value.
- **Value Web**
  - Assuredly not a recipe for success but preliminary conceptions of an emerging form of a fluid and flexible organization.
  - Move from we-do-everything-ourselves unless (value generated by single organization) to we-do-nothing-ourselves-unless (value generated by the network).
- **E-Business Enabled**
  - Especially valid for B2B contexts
  - 5 Representative Business models
  - Tele-working Model:
    - Collaboration using communication technologies
    - Classic example is Electronic Manufacturing Services (Solectron)
  - Virtual Organization Model:
    - Collection of geographically dispersed individuals, groups and organizational units.
    - Example: GeneraL ife (Insurance)
  - Process Outsourcing Model:
    - Example: BPOs, IBM
  - Collaborative Product Development Model
    - Example: Automobile Manufacture -> FORD
  - Value Chain Integration Model:
    - Used to improve communication & collaboration between all supply chain parties.
- **Market Participants**
  - More generic classification of Internet Based Business Models
- **Cyber-mediaries**
  - In disagreement with the widely accepted idea that e-Business will cause industry value chains to be restructured to such an extent that intermediation will no longer be a prominent feature.
  - The real trend might just be towards an increase in intermediation by cyber-mediaries.
    - Organizations which operate in electronic markets to facilitate exchanges between producers and consumers by meeting the needs of both.
    - Directories of Directory Services Intermediaries
    - Virtual Malls
    - Website Evaluators
    - Auditors
    - Spot Market Makers
    - Financial Intermediaries (ESCROW Service for Online purchases).

Note: For details on all e-Business models, Refer Michael Papazoglou, "e-Business – Organizational & Technical Foundations"

e-Business Integration (Patterns)

e-Business Integration occurs in as many forms as there are e-Businesses. At first glance, integration problems and the corresponding solutions are seldom identical. Yet, upon closer examination, you discover that integration solutions can actually be classified into common categories. Each of these categories describes both a "type" of integration problem as well as a solution method. These categories are called integration patterns. Integration patterns help you understand the different methods available to you for a given type of integration problem. They allow you to take a step back and understand the differences in the various scenarios and appreciate the different approaches to integration. Finally, they allow you to view "integration in the big picture." You can learn to break down what may be a complex integration into conceptual categories and understand which technologies to apply.

What Are Integration Patterns?

A pattern is commonly defined as a reliable sample of traits, acts, tendencies, or other observable characteristics. In software development, you may be familiar with the idea of design patterns or process patterns. Design patterns systematically describe object designs that can be employed for a common set of problems. Similarly, process patterns describe proven methods and processes used in software development. In practice, patterns are simply a logical classification of commonly recurring actions, techniques, designs, or organizations. What are integration patterns? Integration patterns emerge from classification of standard solutions for integration scenarios. They are not patterns of design or code. Nor are they patterns of operational processes for an integration project. Instead, each integration pattern defines a type of integration problem, a solution technique, as well as parameters applied for e-Business Integration.

Following are seven common e-Business Integration patterns. They are not meant to be comprehensive, but they cover most of the common integration scenarios implemented today. They encompass both EAI scenarios as well as B2Bi scenarios:

- **EAI (intra-enterprise) Patterns**
  1. Database Replication
  2. Single-Step Application Integration
  3. Multi-Step Application Integration
  4. Brokering Application
- **B2Bi (inter-enterprise) Patterns**
  5. Application-to-Application B2Bi
  6. Data Exchange B2Bi
  7. B2B Process Integration

The EAI Patterns represent patterns commonly applied within a corporate enterprise, whereas the B2Bi Patterns represent the different methods in conducting integrated B2B transactions. The following sections provide a closer look at each of these patterns and discuss some of the details.

Database Replication

The Database Replication pattern may be the most prevalent pattern of EAI integration today. Database replication involves managing copies of data over two or more databases, resulting in redundant data. Companies engage in database replication for numerous reasons. One reason is that many organizations are becoming more distributed in their operations, requiring multiple copies of the same data over several physical locations. Replication is also a means of data recovery. In many organizations, an active secondary database is maintained for data recovery purposes. In the event that the production database needs to be recovered, the secondary replicated database can be used. This also applies for "high availability" systems. In these situations, a redundant copy of "live" data is

maintained to ensure that if the first system is not available, the redundant database system is activated. The two general categories for database replication are synchronous and asynchronous replication.

Single-Step Application Integration

The Single-Step Application Integration (SSAI) pattern extends the asynchronous database replication pattern. Instead of focusing on data consistency between two databases, the SSAI pattern integrates data between applications, moving data from one context to another. It does so by translating data syntax of the source message and reformatting data elements into a new target message. It is "single step" because it requires an intermediary broker to map source messages to target messages. Typically, it is an extension of the asynchronous replication technology, in that it utilizes Message Queuing Middleware such as MQ Series. It is just as likely to be implemented with the less sophisticated FTP in a batch mode. In either case, the point is that it does more than simply move data from point A to point B for consistency's sake. Whereas, in the replication pattern both the source and target data models are likely similar, if not identical at times, this is not necessarily the case for the SSAI pattern. The objective here is not data consistency, but application data integration.

Multi-Step Application Integration

The Multi-Step Application Integration (MSAI) pattern is an extension of the SSAI pattern. MSAI enables the integration of n (source) to m (target) applications. It addresses many-to-many integration, which SSAI cannot, by providing what is known as sequential logical processing. In other words, steps in this pattern are processed sequentially, and rules applied are Boolean logical in nature. Like the single-step pattern, MSAI requires an intermediary to broker the transaction of data between applications. It is often built around an asynchronous event-based system and typically is implemented through the use of Message Queuing Middleware as well. The asynchronous eventbased approach creates loose coupling. Although each system is physically independent, they are logically dependent. In other words, interdependencies exist between the application events that can be expressed in terms of transformations and data integration rules. Data elements from one application can drive the retrieval or processing of messages in another application. The simplest multi-step example in Figure 3.3 involves three applications in which a message from application A is combined with a message from application B that is reformatted for a target application C. It is common for a data element from application A to act as a key to drive the request for information from application B.

Brokering Application

At times integrating two applications is not principally a matter of integrating data, but integrating business logic. The Brokering Application pattern addresses the use of intermediary application logic to link together two or more applications. In plain terms, it means that custom application code is written containing logic to broker interactions between the disparate applications. This custom brokering application sits in the middle as an intermediary for processing requests from different applications

The use of this solution pattern is particularly applicable in the scenarios below:

- Applications Need to Reuse Logic
- Applications Linked by Complex Logic
- Applications Unified Through User Interface

Application-to-Application B2Bi

Now you're ready to move beyond EAI to learn about Application-to-Application B2Bi, extending integration beyond the corporate enterprise. I will describe four additional patterns related specifically to B2B integration, beginning first with the Application-to-Application B2Bi pattern. The Application-to-Application pattern is the logical extension of what occurs in EAI. When EAI vendors tout their products as being B2Bi, this specific pattern is what they have in mind. However, as you will discover, this is not the only pattern and very likely not even the primary pattern for B2Bi. Application-to-Application B2Bi, which is often referred to as inter-enterprise integration, involves corporate

entities linking their applications directly to the applications of their partners or customers. In practice, this type of integration is often implemented as part of a supply chain of goods and services to the customer.

This extension for interenterprise integration means that a number of additional issues need to be accounted for:

- Security
- Federated Control
- Systems Management

Data Exchange B2Bi

The limitation of the Application-to-Application B2Bi pattern is that it can be more demanding to implement. It necessitates that each participant handles and externalizes application native data directly. This makes it difficult to scale the B2B interaction model rapidly when such a demand is placed on the participants. The optimal solution is to provide a rapidly scalable B2Bi model in which participants can exchange data freely with minimal expectation on their infrastructure. The Data Exchange B2Bi pattern enables B2B transactions predicated on a common data exchange format. It is the most widely applied pattern for B2B commerce today. Data Exchange B2Bi is effective because it is simple in concept and has been in use since the days of Electronic Data Interchange (EDI), the forerunner to today's B2B over the Internet.

Although there is a significant incumbency of legacy EDI transactions, the XML-based B2B will ultimately displace EDI as the primary mechanism for e-Business transactions. XML-based data packets are transmitted between two business entities through the use of a data exchange gateway service on both ends. One of the primary responsibilities of the gateway service is to prepare the data packets by placing them within a security envelope. The B2B gateway service supports security standards such as MIME, X.509, and S/Key. It is also responsible for routing data through a standard transport. Most B2B gateway services provide numerous transport options including HTTPS, FTP, and TCP/IP Sockets. However, upon examination, you will find that most B2Bi transactions still deliver XML documents over an HTTPS pipe.

B2B Process Integration

Even with industrywide initiatives such as RosettaNet, a point-to-point data exchange that manages static interactions has some limitations. If Corporation A wants to purchase office supplies from Depot X, it must agree ahead of time on the content of the documents exchanged and buying process. This is, of course, to be expected. However, what if the situation involves managing multiple suppliers or if the interactions become more complex? For instance, a scenario in which suppliers openly bid to compete on pricing will increase the dimensions of process interactions. In that case, managing the B2B transaction is no longer an activity of managing a single point-to-point interaction. Instead, it becomes a challenge of managing business processes that are dynamic rather than static.

The B2B Process Integration pattern takes the limitations raised by the Data Exchange pattern and addresses them by providing Business Process Integration (BPI) services. Just as the Data Exchange pattern allows participants to manage data exchanges dynamically through XML-based documents, the B2B Process Integration pattern allows the participants to manage processes in the same way.

Therefore, richer, more complex relationships can occur between trading partners. B2B Process Integration pattern can be implemented as one of two variations: Closed Process B2Bi or Open Process B2Bi. You might argue that each of these variations constitutes an individual pattern, but because they share the common attribute of being process focused, I have decided to treat them as variations to the B2B Process Integration pattern.

Approaches to Middleware

Middleware is computer software that connects software components or some people and their applications. The software consists of a set of services that allows multiple processes running on one or more machines to interact. This technology evolved to provide for interoperability in support of the move to coherent distributed architectures,

which are most often used to support and simplify complex distributed applications. It includes web servers, application servers, and similar tools that support application development and delivery. Middleware is especially integral to modern information technology based on XML, SOAP, Web services, and service-oriented architecture.

Middleware sits "in the middle" between application software that may be working on different operating systems. It is similar to the middle layer of a three-tier single system architecture, except that it is stretched across multiple systems or applications. Examples include EAI software, telecommunications software, transaction monitors, and messaging-and-queueing software.

The distinction between operating system and middleware functionality is, to some extent, arbitrary. While core kernel functionality can only be provided by the operating system itself, some functionality previously provided by separately sold middleware is now integrated in operating systems. A typical example is the TCP/IP stack for telecommunications, nowadays included in virtually every operating system.

In simulation technology, middleware is generally used in the context of the high level architecture (HLA) that applies to many distributed simulations. It is a layer of software that lies between the application code and the run-time infrastructure. Middleware generally consists of a library of functions, and enables a number of applications—simulations or federates in HLA terminology—to page these functions from the common library rather than re-create them for each application

Definition of Middleware

Software that provides a link between separate software applications. Middleware is sometimes called plumbing because it connects two applications and passes data between them. Middleware allows data contained in one database to be accessed through another. This definition would fit enterprise application integration and data integration software.

ObjectWeb defines middleware as: "The software layer that lies between the operating system and applications on each side of a distributed computing system in a network."

Middleware is computer software that connects software components or applications. The software consists of a set of services that allows multiple processes running on one or more machines to interact. This technology evolved to provide for interoperability in support of the move to coherent distributed architectures, which are most often used to support and simplify complex, distributed applications.

It includes web servers, application servers, and similar tools that support application development and delivery. Middleware is especially integral to modern information technology based on XML, SOAP, Web services, and service-oriented architecture.

In simulation technology, middleware is generally used in the context of the high level architecture (HLA) that applies to many distributed simulations. It is a layer of software that lies between the application code and the run-time infrastructure. Middleware generally consists of a library of functions, and enables a number of applications—simulations or federates in HLA terminology—to page these functions from the common library rather than re-create them for each application.

Origin of Middleware

Middleware is a relatively new addition to the computing landscape. It gained popularity in the 1980s as a solution to the problem of how to link newer applications to older legacy systems, although the term had been in use since 1968. It also facilitated distributed processing, the connection of multiple applications to create a larger application, usually over a network.

Use of middleware

Middleware services provide a more functional set of application programming interfaces to allow an application to (when compared to the operating system and network services.):

- Locate transparently across the network, thus providing interaction with another service or application
- Filter data to make them friendly usable or public via anonymization process for privacy protection (for example)
- Be independent from network services
- Be reliable and always available
- Add complementary attributes like semantics

Middleware offers some unique technological advantages for business and industry. For example, traditional database systems are usually deployed in closed environments where users access the system only via a restricted network or intranet (e.g., an enterprise's internal network). With the phenomenal growth of the World Wide Web, users can access virtually any database for which they have proper access rights from anywhere in the world. Middleware addresses the problem of varying levels of interoperability among different database structures. Middleware facilitates transparent access to legacy database management systems (DBMSs) or applications via a web server without regard to database-specific characteristics .

Businesses frequently use middleware applications to link information from departmental databases, such as payroll, sales, and accounting, or databases housed in multiple geographic locations. In the highly competitive healthcare community, laboratories make extensive use of middleware applications for data mining, laboratory information system (LIS) backup, and to combine systems during hospital mergers. Middleware helps bridge the gap between separate LISs in a newly formed healthcare network following a hospital buyout.

Wireless networking developers can use middleware to meet the challenges associated with wireless sensor network (WSN), or WSN technologies. Implementing a middleware application allows WSN developers to integrate operating systems and hardware with the wide variety of various applications that are currently available.

Middleware can help software developers avoid having to write application programming interfaces (API) for every control program, by serving as an independent programming interface for their applications. For Future Internet network operation through traffic monitoring in multi-domain scenarios, using mediator tools (middleware) is a powerful help since they allow operators, searchers and service providers to supervise Quality of service and analyse eventual failures in telecommunication services.

Finally, e-commerce uses middleware to assist in handling rapid and secure transactions over many different types of computer environments. In short, middleware has become a critical element across a broad range of industries, thanks to its ability to bring together resources across dissimilar networks or computing platforms.

Types of middleware

Hurwitz's classification system organizes the many types of middleware that are currently available. These classifications are based on scalability and recoverability:

- Remote Procedure Call — Client makes calls to procedures running on remote systems. Can be asynchronous or synchronous.
- Message Oriented Middleware — Messages sent to the client are collected and stored until they are acted upon, while the client continues with other processing.
- Object Request Broker — This type of middleware makes it possible for applications to send objects and request services in an object-oriented system.
- SQL-oriented Data Access — middleware between applications and database servers.

- Embedded Middleware — communication services and integration interface software/firmware that operates between embedded applications and the real time operating system.
- Other sources include these additional classifications:
- Transaction processing monitors — Provides tools and an environment to develop and deploy distributed applications.
- Application servers — software installed on a computer to facilitate the serving (running) of other applications.
- Enterprise Service Bus — An abstraction layer on top of an Enterprise Messaging System.

RPC

In computer science, a remote procedure call (RPC) is an inter-process communication that allows a computer program to cause a subroutine or procedure to execute in another address space (commonly on another computer on a shared network) without the programmer explicitly coding the details for this remote interaction. That is, the programmer writes essentially the same code whether the subroutine is local to the executing program, or remote. When the software in question uses object-oriented principles, RPC is called remote invocation or remote method invocation. Note that there are many different (often incompatible) technologies commonly used to accomplish this.

History and origins

The idea of RPC (Remote Procedure Call) goes back at least as far as 1976, when it was described in RFC 707. One of the first business uses of RPC was by Xerox under the name "Courier" in 1981. The first popular implementation of RPC on Unix was Sun's RPC (now called ONC RPC), used as the basis for NFS (Sun). Another early Unix implementation was Apollo Computer's Network Computing System (NCS). NCS later was used as the foundation of DCE/RPC in the OSF's Distributed Computing Environment (DCE). A decade later Microsoft adopted DCE/RPC as the basis of the Microsoft RPC (MSRPC) mechanism, and implemented DCOM on top of it. Around the same time (mid-90's), Xerox PARC's ILU, and the Object Management Group's CORBA, offered another RPC paradigm based on distributed objects with an inheritance mechanism.

Message passing

An RPC is initiated by the client, which sends a request message to a known remote server to execute a specified procedure with supplied parameters. The remote server sends a response to the client, and the application continues its process. There are many variations and subtleties in various implementations, resulting in a variety of different (incompatible) RPC protocols. While the server is processing the call, the client is blocked (it waits until the server has finished processing before resuming execution). An important difference between remote procedure calls and local calls is that remote calls can fail because of unpredictable network problems. Also, callers generally must deal with such failures without knowing whether the remote procedure was actually invoked. Idempotent procedures (those that have no additional effects if called more than once) are easily handled, but enough difficulties remain that code to call remote procedures is often confined to carefully written low-level subsystems.

The steps in making a RPC

1. The client calling the Client stub. The call is a local procedure call, with parameters pushed on to the stack in the normal way.
2. The client stub packing the parameters into a message and making a system call to send the message. Packing the parameters is called marshaling.
3. The kernel sending the message from the client machine to the server machine.
4. The kernel passing the incoming packets to the server stub.
5. Finally, the server stub calling the server procedure. The reply traces the same in other direction.

Standard contact mechanisms

To let different clients access servers, a number of standardized RPC systems have been created. Most of these use an interface description language (IDL) to let various platforms call the RPC. The IDL files can then be used to generate code to interface between the client and server. The most common tool used for this is RPCGEN.

Other RPC analogues

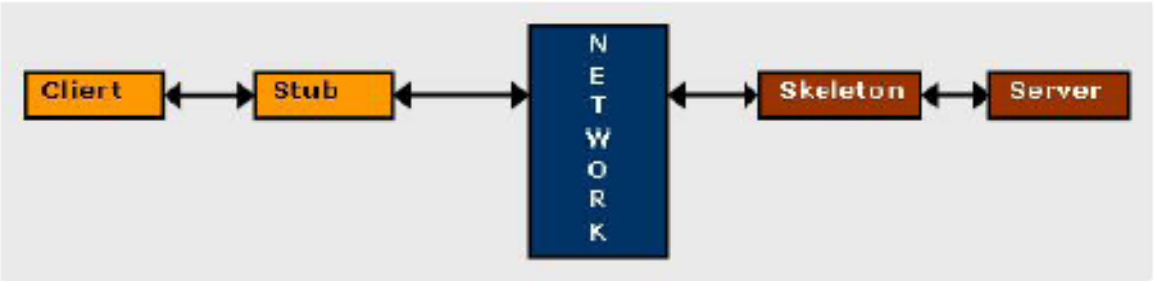
RPC analogues found elsewhere:

- Java's Java Remote Method Invocation (Java RMI) API provides similar functionality to standard UNIX RPC
- methods.
- Modula-3's Network Objects, which were the basis for Java's RMI
- XML-RPC is an RPC protocol that uses XML to encode its calls and HTTP as a transport mechanism.
- Microsoft .NET Remoting offers RPC facilities for distributed systems implemented on the Windows platform.
- RPyC implements RPC mechanisms in Python, with support for asynchronous calls.
- Pyro Object Oriented form of RPC for Python.
- Etch (protocol) framework for building network services.
- Facebook's Thrift protocol and framework.
- CORBA provides remote procedure invocation through an intermediate layer called the "Object Request Broker"
- DRb allows Ruby programs to communicate with each other on the same machine or over a network. DRb uses
- remote method invocation (RMI) to pass commands and data between processes.
- AMF allows Flex applications to communicate with back-ends or other applications that support AMF.
- Libevent provides a framework for creating RPC servers and clients.
- Windows Communication Foundation is an application

RMI

The Java Remote Method Invocation Application Programming Interface (API), or Java RMI, is a Java application programming interface that performs the object-oriented equivalent of remote procedure calls (RPC).

1. The original implementation depends on Java Virtual Machine (JVM) class representation mechanisms and it thus only supports making calls from one JVM to another. The protocol underlying this Java-only implementation is known as Java Remote Method Protocol (JRMP).
2. In order to support code running in a non-JVM context, a CORBA version was later developed.



A typical implementation model of Java-RMI using stub and skeleton objects. Java 2 SDK, Standard Edition, v1.2 removed the need for a skeleton.

Usage of the term RMI may denote solely the programming interface or may signify both the API and JRMP, whereas the term RMI-IIOP (read: RMI over IIOP) denotes the RMI interface delegating most of the functionality to the supporting CORBA implementation.

The programmers of the original RMI API generalized the code somewhat to support different implementations, such as a HTTP transport. Additionally, the ability to pass arguments "by value" was added to CORBA in order to support the RMI interface. Still, the RMI-IIOP and JRMP implementations do not have fully identical interfaces.

RMI functionality comes in the package java.rmi, while most of Sun's implementation is located in the sun.rmi package. Note that with Java versions before Java 5.0 developers had to compile RMI stubs in a separate compilation step using rmic. Version 5.0 of Java and beyond no longer require this step.

Jini offers a more advanced version of RMI in Java. It functions similarly but provides more advanced searching capabilities and mechanisms for distributed object applications.

Example

The following classes implement a simple client-server program using RMI that displays a message. **RmiServer class—Listens to RMI requests and implements the interface which is used by the client to invoke remote methods.**

```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.RMISecurityManager;
import java.rmi.server.UnicastRemoteObject;
import java.rmi.registry.*;
public class RmiServer extends UnicastRemoteObject
implements RmiServerIntf
{
static public final String MESSAGE = "Hello world";
public RmiServer() throws RemoteException
{
super();
}
public String getMessage()
{
return MESSAGE;
}
public static void main(String args[])
{
System.out.println("RMI server started");
// Create and install a security manager
if (System.getSecurityManager() == null)
{
System.setSecurityManager(new RMISecurityManager());
System.out.println("Security manager installed.");
}
else
System.out.println("Security manager already exists.");
try //special exception handler for registry creation
{
LocateRegistry.createRegistry(1099);
System.out.println("java RMI registry created.");
}
catch (RemoteException e)
{
//do nothing, error means registry already exists
System.out.println("java RMI registry already exists.");
}
try
{
```

```
//Instantiate RmiServer
RmiServer obj = new RmiServer();
// Bind this object instance to the name "RmiServer"
Naming.rebind("//localhost/RmiServer", obj);
System.out.println("PeerServer bound in registry");
}
catch (Exception e)
{
System.err.println("RMI server exception:");
e.printStackTrace();
}
}
}
```

RmiServerIntf class—Defines the interface that is used by the client and implemented by the server.

```
import java.rmi.Remote;
import java.rmi.RemoteException;
public interface RmiServerIntf extends Remote
{
public String getMessage() throws RemoteException;
}
```

RmiClient class—This is the client which gets the reference to the remote object and invokes its method to get a message.

```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.RMISecurityManager;
public class RmiClient
{
// "obj" is the reference of the remote object
RmiServerIntf obj = null;
public String getMessage()
{
try
{
obj =
(RmiServerIntf)Naming.lookup("//localhost/RmiServer");
return obj.getMessage();
}
catch (Exception e)
{
System.err.println("RmiClient exception: " +
e.getMessage());
e.printStackTrace();
return e.getMessage();
}
}
public static void main(String args[])
{
// Create and install a security manager
if (System.getSecurityManager() == null)
{
System.setSecurityManager(new RMISecurityManager());
}
RmiClient cli = new RmiClient();
System.out.println(cli.getMessage());
}
}
```

Before running this subj, we need to make 'Stub' file of interface we used. For this task we have RMI compiller - 'rmic'

- **Note: we make stub file from *.class with implementation remote interface, not '*.java'***

```
rmic RmiServer
```

server.policy—This file is required on the server to allow TCP/IP communication for the remote registry and for the RMI server.

```
grant {
permission java.net.SocketPermission "127.0.0.1:*",
"connect,resolve";
permission java.net.SocketPermission "127.0.0.1:*", "accept";
};
```

The server.policy file should be used using the D switch of Java RTE, e.g.:

java.exe -Djava.security.policy=server.policy RmiServer

client.policy—This file is required on the client to connect to RMI Server using TCP/IP.

```
grant {
permission java.net.SocketPermission "127.0.0.1:*",
"connect,resolve";
};
```

no.policy—Also if you have a troubles with connecting, try this file for server or client.

```
grant {
permission java.security.AllPermission;
};
```

Enterprise Application Integration

Enterprise Application Integration (EAI) is defined as the use of software and computer systems architectural principles to integrate a set of enterprise computer applications. Enterprise Application Integration (EAI) is an integration framework composed of a collection of technologies and services which form a middleware to enable integration of systems and applications across the enterprise. Supply chain management applications (for managing inventory and shipping), customer relationship management applications (for managing current and potential customers), business intelligence applications (for finding patterns from existing data from operations), and other types of applications (for managing data such as human resources data, health care, internal communications, etc) typically cannot communicate with one another in order to share data or business rules.

For this reason, such applications are sometimes referred to as islands of automation or information silos. This lack of communication leads to inefficiencies, wherein identical data are stored in multiple locations, or straightforward processes are unable to be automated. Enterprise application integration (EAI) is the process of linking such applications within a single organization together in order to simplify and automate business processes to the greatest extent possible, while at the same time avoiding having to make sweeping changes to the existing applications or data structures. In the words of the Gartner Group, EAI is the “unrestricted sharing of data and business processes among any connected application or data sources in the enterprise.”

One large challenge of EAI is that the various systems that need to be linked together often reside on different operating systems, use different database solutions and different computer languages, and in some cases are legacy systems that are no longer supported by the vendor who originally created them. In some cases, such systems are dubbed "stovepipe systems" because they consist of components that have been jammed together in a way that makes it very hard to modify them in any way.

Purposes of EAI

EAI can be used for different purposes:

- Data (information) Integration: Ensuring that information in multiple systems is kept consistent. This is also known as EII (Enterprise Information Integration).
- Vendor independence: Extracting business policies or rules from applications and implementing them in the EAI system, so that even if one of the business applications is replaced with a different vendor's application, the business rules do not have to be re-implemented.
- Common Facade: An EAI system could front-end a cluster of applications, providing a single consistent access interface to these applications and shielding users from having to learn to interact with different software packages.

EAI patterns

Integration patterns

There are two patterns that EAI systems implement:

- Mediation: Here, the EAI system acts as the go-between or broker between (interface or communicating) multiple applications. Whenever an interesting event occurs in an application (e. g., new information created, new transaction completed, etc.) an integration module in the EAI system is notified. The module then propagates the changes to other relevant applications.
- Federation: In this case, the EAI system acts as the overarching facade across multiple applications. All event calls from the 'outside world' to any of the applications are front-ended by the EAI system. The EAI system is configured to expose only the relevant information and interfaces of the underlying applications to the outside world, and performs all interactions with the underlying applications on behalf of the requester.

Both patterns are often used concurrently. The same EAI system could be keeping multiple applications in sync (mediation), while servicing requests from external users against these applications (federation).

Access patterns

EAI supports both asynchronous and synchronous access patterns, the former being typical in the mediation case and the latter in the federation case.

Lifetime patterns

An integration operation could be short-lived (e. g., keeping data in sync across two applications could be completed within a second) or long-lived (e. g., one of the steps could involve the EAI system interacting with a human work flow application for approval of a loan that takes hours or days tocomplete).

EAI topologies

There are two major topologies: hub-and-spoke, and bus. Each has its own advantages and disadvantages. In the hub-and-spoke model, the EAI system is at the center (the hub), and interacts with the applications via the spokes. In the bus model, the EAI system is the bus (or is implemented as a resident module in an already existing message bus or message-oriented middleware).

Technologies

Multiple technologies are used in implementing each of the components of the EAI system:

Bus/hub

This is usually implemented by enhancing standard middleware products (application server, message bus) or implemented as a stand-alone program (i. e., does not use any middleware), acting as its own middleware.

Application connectivity

The bus/hub connects to applications through a set of adapters (also referred to as connectors). These are programs that know how to interact with an underlying business application. The adapter performs two-way communication, performing requests from the hub against the application, and notifying the hub when an event of interest occurs in the application (a new record inserted, a transaction completed, etc.). Adapters can be specific to an application (e. g., built against the application vendor's client libraries) or specific to a class of applications (e. g., can interact with any application through a standard communication protocol, such as SOAP or SMTP). The adapter could reside in the same process space as the bus/hub or execute in a remote location and interact with the hub/bus through industry standard protocols such as message queues, web services, or even use a proprietary protocol. In the Java world, standards such as JCA allow adapters to be created in a vendor-neutral manner.

Data format and transformation

To avoid every adapter having to convert data to/from every other applications' formats, EAI systems usually stipulate an application-independent (or common) data format. The EAI system usually provides a data transformation service as well to help convert between application-specific and common formats. This is done in two steps: the adapter converts information from the application's format to the bus's common format. Then, semantic transformations are applied on this (converting zip codes to city names, splitting/merging objects from one application into objects in the other applications, and so on).

Integration modules

An EAI system could be participating in multiple concurrent integration operations at any given time, each type of integration being processed by a different integration module. Integration modules subscribe to

events of specific types and process notifications that they receive when these events occur. These modules could be implemented in different ways: on Java-based EAI systems, these could be web applications or EJBs or even POJOs that conform to the EAI system's specifications.

Support for transactions

When used for process integration, the EAI system also provides transactional consistency across applications by executing all integration operations across all applications in a single overarching distributed transaction (using two-phase commit protocols or compensating transactions).

Communication architectures

Currently, there are many variations of thought on what constitutes the best infrastructure, component model, and standards structure for Enterprise Application Integration. There seems to be consensus that four components are essential for modern enterprise application integration architecture:

1. A centralized broker that handles security, access, and communication. This can be accomplished through integration servers (like the School Interoperability Framework (SIF) Zone Integration Servers) or through similar software like the Enterprise service bus (ESB) model that acts as a SOAP-oriented services manager.
2. An independent data model based on a standard data structure, also known as a Canonical data model. It appears that XML and the use of XML style sheets has become the de facto and in some cases de jure standard for this uniform business language.
3. A connector or agent model where each vendor, application, or interface can build a single component that can speak natively to that application and communicate with the centralized broker.
4. A system model that defines the APIs, data flow and rules of engagement to the system such that components can be built to interface with it in a standardized way.

Although other approaches like connecting at the database or user-interface level have been explored, they have not been found to scale or be able to adjust. Individual applications can publish messages to the centralized broker and subscribe to receive certain messages from that broker. Each application only requires one connection to the broker. This central control approach can be extremely scalable and highly evolvable. Enterprise Application Integration is related to middleware technologies such as message-oriented middleware (MOM), and data representation technologies such as XML. Other EAI technologies involve using web services as part of service-oriented architecture as a means of integration. Enterprise Application Integration tends to be data centric. In the near future, it will come to include content integration and business processes.

EAI Implementation Pitfalls

In 2003 it was reported that 70% of all EAI projects fail.

Most of these failures are not due to the software itself or technical difficulties, but due to management issues. Integration Consortium European Chairman Steve Craggs has outlined the seven main pitfalls undertaken by companies using EAI systems and explains solutions to these problems.

- **Constant change**

The very nature of EAI is dynamic and requires dynamic project managers to manage their implementation.

- **Shortage of EAI experts**

EAI requires knowledge of many issues and technical aspects.

- **Competing standards**

Within the EAI field, the paradox is that EAI standards themselves are not universal.

• **EAI is a tool paradigm**

EAI is not a tool, but rather a system and should be implemented as such.

• **Building interfaces is an art**

Engineering the solution is not sufficient. Solutions need to be negotiated with user departments to reach a common consensus on the final outcome. A lack of consensus on interface designs leads to excessive effort to map between various systems data requirements.

• **Loss of detail**

Information that seemed unimportant at an earlier stage may become crucial later.

• **Accountability**

Since so many departments have many conflicting requirements, there should be clear accountability for the system's final structure.

Other potential problems may arise in these areas:

• **Emerging Requirements**

EAI implementations should be extensible and modular to allow for future changes.

• **Protectionism**

The applications whose data is being integrated often belong to different departments that have technical, cultural, and political reasons for not wanting to share their data with other departments

Advantages and Disadvantages

- Advantages
  - Real time information access among systems
  - Streamlines business processes and helps raise organizational efficiency
  - Maintains information integrity across multiple systems
  - Ease of development and maintenance
- Disadvantages
  - High initial development costs, especially for small and mid-sized businesses (SMBs)
  - Require a fair amount of up front business design, which many managers are not able to envision or not willing to invest in. Most EAI projects usually start off as point-to-point efforts, very soon becoming unmanageable as the number of applications increase

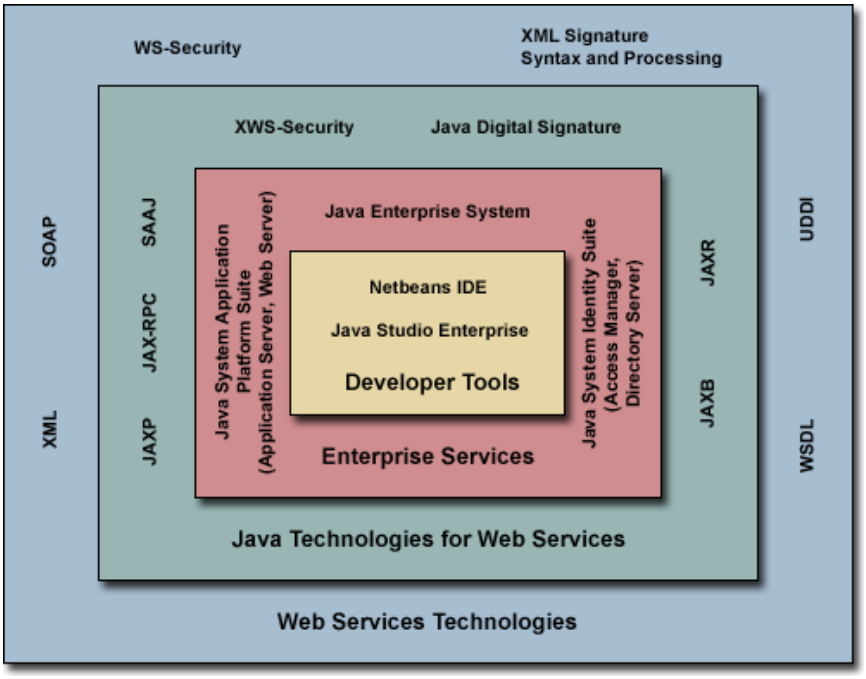
Software as a Service (SaaS)

Software as a Service is a relatively new trend among software providers in which an application is designed to be accessed on the Web rather than installed on the user's workstation. Though the software is Web-based, it is not browser-based. Instead, the software provider develops a thin client application that the user can download one time to their workstation. Once the application is installed, the user's workstation maintains constant communication with the software provider's server whenever the application is in use. The application is called a thin client because it provides a considerably smaller download than locally housed applications, or fat clients, while maintaining the same robust functionality. The hosted server houses all of the user's data; none is stored on the user's computer. There are many benefits to using hosted software.

SOA

Service-oriented architecture (SOA) is a hot topic in enterprise computing because many IT professionals see the potential of an SOA -- especially a web services-based SOA -- in dramatically speeding up the application development process. They also see it as a way to build applications and systems that are more adaptable, and in doing so, they see IT becoming more agile in responding to changing business needs. Not only is SOA a hot topic, but it's clearly the wave of the future. Gartner reports that "By 2008, SOA will be a prevailing software engineering practice, ending the 40-year domination of monolithic software architecture" and that "Through 2008, SOA and web services will be implemented together in more than 75 percent of new SOA or web services projects."

But despite this strong trend, some in the IT community don't feel that the web services underpinning for an SOA is mature enough for their enterprise to consider migration to a service-oriented architecture. For others, the terms service-oriented architecture and web services draw a blank stare. An earlier article, presented a brief overview of SOA and the role of web services in realizing it. This article supplements that earlier article. If you're not familiar with SOA and web services, this article aims to familiarize you with them. It defines some of the key terms and concepts related to SOA and web services. A critical mass of widely-adopted technologies is available now to implement and use a web services-based SOA, and more technologies, as well as tools, are on the way. Figure 1 identifies some of these technologies and tools. Each layer of the figure shows technologies or tools that leverage technologies in the surrounding layers.



A service-oriented architecture is an information technology approach or strategy in which applications make use of (perhaps more accurately, rely on) services available in a network such as the World Wide Web. Implementing a service-oriented architecture can involve developing applications that use services, making applications available as services so that other applications can use those services, or both. A service provides a specific function, typically a business function, such as analyzing an individual's credit history or processing a purchase order. A service can provide a single discrete function, such as converting one type of currency into another, or it can perform a set of related business functions, such as handling the various operations in an airline reservations system. Services that perform a related set of business functions, as opposed to a single function, are said to be "coarse grained." Multiple services can be used together in a coordinated way. The aggregated, or composite, service can be used to satisfy a more complex business requirement. In fact, one way of looking at an SOA is as an approach to connecting applications (exposed as services) so that they can communicate with (and take advantage of) each other. In other words, a serviceoriented architecture is a way of sharing functions (typically business functions) in a widespread and flexible way.

The concept of an SOA is not new. Service-oriented architectures have been used for years. What distinguishes an SOA from other architectures is loose coupling. Loose coupling means that the client of a service is essentially independent of the service. The way a client (which can be another service) communicates with the service doesn't depend on the implementation of the service. Significantly, this means that the client doesn't have to know very much about the service to use it. For instance, the client doesn't need to know what language the service is coded in or what platform the service runs on. The client communicates with the service according to a specified, well-defined interface, and then leaves it up to the service implementation to perform the necessary processing. If the implementation of the service changes, for instance, the airline reservations application is revised, the client communicates with it in the same way as before, provided that the interface remains the same. Loose coupling enables services to be documentoriented (or document-centric). A document-oriented service accepts a document as input, as opposed to something more granular like a numeric value or Java object. The client does not know or care what business function in the service will process the document. It's up to the service to determine what business function (or functions) to apply based on the content of the document. However what is relatively new is the emergence of web services-based SOAs. A web service is a service that communicates with clients through a set of standard protocols and technologies. These web services standards are implemented in platforms and products from all the major software vendors, making it possible for clients and services to communicate in a consistent way across a wide spectrum of platforms and operating environments. This universality has made web services the most prevalent approach to implementing an SOA.

Optionally, an SOA can also include a service that provides a directory or registry of services. The registry contains information about the service such as it's interface. A client can discover services by examining the registry. A registry can also be coupled with a repository component that stores additional information about each service. This additional "metadata" can include business process information such as policy statements.

Why SOA

There are many reasons for an enterprise to take an SOA approach, and more specifically, a web servicesbased SOA approach. Some of the primary reasons are:

Reusability: What drives the move to SOA is reuse of business services. Developers within an enterprise and across enterprises (particularly, in business partnerships) can take the code developed for existing business applications, expose it as web services, and then reuse it to meet new business requirements. Reusing functionality that already exists outside or inside an enterprise instead of developing code that reproduces those functions can result in a huge savings in application development cost and time. The benefit of reuse grows dramatically as more and more business services get built, and incorporated into different applications. A major obstacle in taking advantage of existing code is the uniqueness of specific applications and systems. Typically, solutions developed in different enterprises, even different departments within the same enterprise, have unique characteristics. They run in different operating environments, they're coded in different languages, they use different programming interfaces and protocols. You need to understand how and where these applications and systems run to communicate with them. The work involved in doing this analysis and the development effort in tying these pieces together can be very time consuming. Witness the pain IT organizations generally encounter when they try to integrate their applications with systems from business partners (or even with legacy systems from other parts of their own company). In an SOA, the only characteristic of a service that a requesting application needs to know about is the public interface. The functions of an application or system (including legacy systems) can be dramatically easier to access as a service in an SOA than in some other architecture. So integrating applications and systems can be much simpler.

Interoperability: The SOA vision of interaction between clients and loosely-coupled services means widespread interoperability. In other words, the objective is for clients and services to communicate and understand each other no matter what platform they run on. This objective can be met only if clients and services have a standard way of communicating with each other -- a way that's consistent across platforms, systems, and languages. In fact, web services provide exactly that. Web services comprise a maturing set of protocols and technologies that are widely

accepted and used, and that are platform, system, and language independent. In addition, these protocols and technologies work across firewalls, making it easier for business partners to share vital services. Promising to make things even more consistent is the WS-I basic profile, introduced by the Web Services Interoperability Organization (an organization chartered to promote web services interoperability). The WS-I basic profile identifies a core set of web services technologies that when implemented in different platforms and systems, helps ensure that services on these different platforms and systems, and written in different languages, can communicate with each other. The WS-I basic profile has widespread backing in the computer industry, virtually guaranteeing interoperability of services that conform to the profile.

Scalability: Because services in an SOA are loosely coupled, applications that use these services tend to scale easily -- certainly more easily than applications in a more tightly-coupled environment. That's because there are few dependencies between the requesting application and the services it uses. The dependencies between client and service in a tightly-coupled environment are compounded (and the development effort made significantly more complex) as an application that uses these services scales up to handle more users. Services in a web services-based SOA tend to be coarse-grained, document-oriented, and asynchronous. As mentioned earlier, coarse-grained services offer a set of related business functions rather than a single function. For example, a coarse-grained service might handle the processing of a complete purchase order. By comparison, a fine-grained service might handle only one operation in the purchase order process. Again, as mentioned earlier, a document-oriented service accepts a document as input, as opposed to something more granular like a numeric value or Java object. An example of a document-oriented service might be a travel agency service that accepts as input a document that contains travel information for a specific trip request. An asynchronous service performs its processing without forcing the client to wait for the processing to finish. A synchronous service forces the client to wait. The relatively limited interaction required for a client to communicate with a coarse-grained, asynchronous service, especially a service that handles a document such as a purchase order, allows applications that use these services to scale without putting a heavy communication load on the network.

Flexibility: Loosely-coupled services are typically more flexible than more tightly-coupled applications. In a tightly-coupled architecture, the different components of an application are tightly bound to each other, sharing semantics, libraries, and often sharing state. This makes it difficult to evolve the application to keep up with changing business requirements. The loosely-coupled, document-based, asynchronous nature of services in an SOA allows applications to be flexible, and easy to evolve with changing requirements.

Cost Efficiency: Other approaches that integrate disparate business resources such as legacy systems, business partner applications, and department-specific solutions are expensive because they tend to tie these components together in a customized way. Customized solutions are costly to build because they require extensive analysis, development time, and effort. They're also costly to maintain and extend because they're typically tightly-coupled, so that changes in one component of the integrated solution require changes in other components. A standards-based approach such as a web services-based SOA should result in less costly solutions because the integration of clients and services doesn't require the in-depth analysis and unique code of customized solutions. Also, because services in an SOA are loosely-coupled, applications that use these services should be less costly to maintain and easier to extend than customized solutions. In addition, a lot of the Web-based infrastructure for a web services-based SOA is already in place in many enterprises, further limiting the cost. Last, but not least, SOA is about reuse of business functions exposed as coarse-grained services. This is potentially the biggest cost saving of all.

SOAP

Most people tend to think of SOAP as nothing more than a protocol for Remote Procedure Calls (RPC) over Hypertext Transfer Protocol (HTTP). However, this is only one implementation of SOAP, which is defined as a lightweight protocol for passing structured and typed data between two peers using XML. The specification doesn’t require the use of HTTP or even a request/response type of conversation. Instead, SOAP can be used with any protocol that

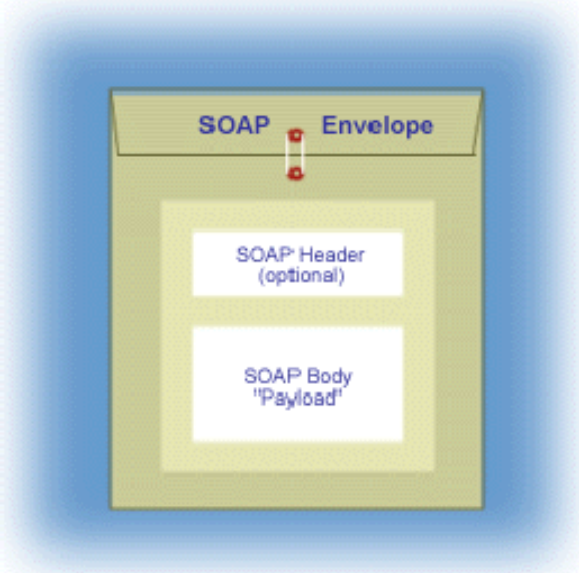
supports the transmission of XML data from a sender to a receiver. In fact, both Microsoft and IBM have implemented SOAP messages over SMTP, which means SOAP messages can be routed through e-mail servers.

The bottom line is SOAP is nothing more than a lightweight messaging protocol, which can be used to send messages between peers. The main goals of SOAP are focused on providing a common way to package message data and define encoding rules used to serialize and deserialize the data during transmission. Another goal was to provide a model that can be used to implement RPC operations using SOAP. All these goals are considered "orthogonal" in the specification, which means they are independent, but related. For example: A SOAP message should define encoding rules, but these rules needn't be the same rules defined in the specification.

Example:

```
<SOAP-ENV: Envelope
xmlns:SOAP-ENV=
"http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:
encodingStyle=
"http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Header>
<t:Transaction xmlns:t="some-URI">
SOAP-ENV:mustUnderstand="1">
</t:Transaction>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
<m:GetBookPrice xmlns:m="some-URI">
<title>My Life and Times</title>
<author>Felix Harrison</author>
</m: GetBookPrice>
</SOAP-ENV:Body>
</SOAP-Envelope>
```

A related standard, SOAP Messages With Attachments, specifies the format of a SOAP message that includes attachments (such as, images). SOAP messages (with or without attachments) are independent of any operating system or platform and can be transported using a variety of communication protocols, such as HTTP or SMTP. WS-I Basic Profile 1.0 references SOAP 1.1.



WSDL

How does a client know what format to use in making a request to a service? For that matter, how do the client and the service know what the request means? The answers to these questions are provided by information in an XML

document, called a WSDL document, that contains a description of the web service's interface. A WSDL document contains information specified in Web Service Description Language (WSDL), as defined in the WSDL specification. WSDL defines an XML schema for describing a web service. To uncover the description for a Web service, a client needs to find the service's WSDL document. One way, perhaps the most typical way, to do this is for the client to find a pointer to the WSDL document in the web service's registration, which can be in a UDDI registry or an ebXML registry/repository. A typical scenario is that a business registers its service. The registry entry includes a pointer to a WSDL file that contains the WSDL document for the service. Another business searches the registry and finds the service. A programmer uses the interface information in the WSDL document to construct the appropriate calls to the service. A WSDL document describes a web service as a collection of abstract items called "ports" or "endpoints." A WSDL document also defines the actions performed by a web service and the data transmitted to these actions in an abstract way. Actions are represented by "operations," and data is represented by "messages." A collection of related operations is known as a "port type." A port type constitutes the collection of actions offered by a web service. What turns a WSDL description from abstract to concrete is a "binding." A binding specifies the network protocol and message format specifications for a particular port type. A port is defined by associating a network address with a binding. If a client locates a WSDL document and finds the binding and network address for each port, it can call the service's operations according to the specified protocol and message format.

Here, for example, is a WSDL document for an online book search service. Notice in the example that the WSDL document specifies one operation for the service: getBooks:

```
<operation name="getBooks" ...
```

The input message for the operation, BookSearchInput contains a string value named isbn:

```
<complexType>
<all>
<element name="isbn"
type="string"/>
</all>
```

The output message for the operation, BookSearchOutput returns a string value named title:

```
<complexType>
<all>
<element name="title"
type="string"/>
</all>
```

Notice too that the WSDL document specifies a SOAP protocol binding. The style attribute in the binding element specifies that the receiver should interpret the payload in the SOAP message as an RPC method call:

```
<soap:binding
transport="transport=http://schemas.xmlsoap.org/soap/http"
style="rpc"/>
```

Alternatively, the style attribute in a binding element could specify "document". In that case, a complete document (typically an XML document) would be exchanged in the call. In fact, the document style of binding is more typical of services in an SOA. One of the advantages of passing an XML document to a service instead of an RPC-style message is that it tends to be easier to validate and to apply business rules. The WS-I Basic Profile 1.0 specifies the constructs in WSDL 1.1 that be can be used to ensure interoperability. It also clarifies some of the construct descriptions in the WSDL 1.1 specification.

UDDI and ebXML

As mentioned earlier, an SOA can also include a service that provides a directory or registry of services. But how can a service be described in a registry so that a program looking for that service can easily find it and understand what it

does? The Universal Description, Discovery, and Integration (UDDI) specifications define how to publish and discover information about services in a UDDI-conforming registry. More specifically, the specifications define a UDDI schema and a UDDI API. The UDDI schema identifies the types of XML data structures that comprise an entry in the registry for a service. Think of a UDDI registry as a "Yellow Pages" for web services. Like the Yellow Pages directory for phone numbers, a UDDI registry provides information about a service such as the name of the service, a brief description of what it does, an address where the service can be accessed, and a description of the interface for accessing the service. The accompanying figure illustrates the schema. It shows the five types of XML data structures that comprise a registration.

Here is an example that shows part of a complete BusinessEntity structure for a hypothetical company named BooksToGo. The company provides various web services, including an online book ordering service.

```
<businessEntity businessKey="35AF7F00-1419-11D6-A0DC-000C0E00ACDD"
authorizedName="0100002CAL"
operator="www-3.ibm.com/services/uddi">
<name>BooksToGo</name>
<description xml:lang="en">
The source for all professional books
</description>
<contacts>
<contact>
<personName>Benjamin Boss</personName>
<phone>
(877)1111111
</phone>
</contact>
</contacts>
```

The API describes the SOAP messages that are used to publish an entry in a registry, or discover an entry in a registry. Here, for example, is a message that searches for all business entities in a registry whose name begins with the characters "Books":

```
<find_business generic="2.0" xmlns=um:uddi-org:api-v2">
<name>Books%</name>
</find_business>
```

UDDI 2.0 is part of the WS-I Basic Profile 1.1

WS-Security

WS-Security is a standard released by OASIS in March 2004. It describes security-related enhancements to SOAP messaging that provide for message integrity and confidentiality. Integrity means that a SOAP message is not tampered with as it travels from a client to its final destination. Confidentiality means that a SOAP message is only seen by intended recipients. WS-Security uses security tokens to enable SOAP message security and integrity. A security token is a collection of claims made by the sender of a SOAP message (such as the sender's identity). A sender is authenticated by combining a security token with a digital signature – the signature is used as proof that the sender is indeed associated with the security token. The standard also provides a general-purpose mechanism for associating security tokens with messages, and describes how to encode binary security tokens. WS-Security is quite flexible and can be used with a wide variety of security models and encryption technologies, such as Public-key infrastructure (PKI) and Kerberos, as well as the Secure Socket Layer (SSL)/ Transport Layer Security (TLS) protocol that provides basic end-to-end security communication on the Internet.

Example:

```
(1)    <?xml version="1.0" encoding="utf-8"?>
(2)     <S:Envelope xmlns:S="http://www.w3.org/2001/12/soap-envelope"
            xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
(3)       <S:Header>
```

```
(4)          <m:path xmlns:m="http://schemas.xmlsoap.org/rp/">
(5)            <m:action>http://fabrikam123.com/getQuote</m:action>
(6)            <m:to>http://fabrikam123.com/stocks</m:to>
(007)           <m:id>uuid:84b9f5d0-33fb-4a81-b02b-5b760641c1d6</m:id>
(8)          </m:path>
(9)          <wsse:Security
              xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext">
(10)           wsse:UsernameToken Id="MyID">
(11)               <wsse:Username>Zoe</wsse:Username>
(12)           </wsse:UsernameToken>
(13)           <ds:Signature>
(14)              <ds:SignedInfo>
(15)                <ds:CanonicalizationMethod
                    Algorithm=
                           "http://www.w3.org/2001/10/xml-exc-c14n#"/>
(16)                 <ds:SignatureMethod
                         Algorithm=
                         "http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
(17)                 <ds:Reference URI="#MsgBody">
(18)                    <ds:DigestMethod
                           Algorithm=
                         "http://www.w3.org/2000/09/xmldsig#sha1"/>
(19)                    <ds:DigestValue>LyLsF0Pi4wPU...</ds:DigestValue>
(20)                 </ds:Reference>
(21)              </ds:SignedInfo>
(22)              <ds:SignatureValue>DJbchm5gK...</ds:SignatureValue>
(23)              <ds:KeyInfo>
(24)                  <wsse:SecurityTokenReference>
(25)                   <wsse:Reference URI="#MyID"/>
(26)                  </wsse:SecurityTokenReference>
(27)              </ds:KeyInfo>
(28)           </ds:Signature>
(29)        </wsse:Security>
(30)     </S:Header>
(31)     <S:Body Id="MsgBody">
(32)       <tru:StockSymbol xmlns:tru="http://fabrikam123.com/payloads">
               QQQ
           </tru:StockSymbol>
(33)     </S:Body>
(34)  </S:Envelope>
```

The lines in the example that illustrate WS-Security enhancements are:

- Line (009). This specifies a <Security> header that contains security information for an intended receiver. This element continues until line (029).
- Lines (010) to (012). These lines specify a security token that is associated with the message. In this case, it defines the username of the client using the <UsernameToken> element. The security token can also be used to specify a password. However in this example it's assumed that the receiver knows the password - in other words, it is a shared secret.
- Lines (013) to (028) specify a digital signature. This signature ensures the integrity of the signed elements (that they aren't modified). The signature follows the XML Signature specification. In this example, the signature is based on a key generated from the user's password. Typically, stronger signing mechanisms would be used.
- Lines (014) to (021) describe the digital signature. Line (015) specifies how to canonicalize (normalize) the data that is being signed.
- Lines (017) to (020) select the elements that are signed. Specifically, line (017) indicates that the <S:Body> element is signed. In this example, only the message body is signed. Typically, additional elements of the message, such as parts of the routing header, should be included in the signature.

- Line (022) specifies the signature value of the canonicalized form of the data that is being signed as defined in the XML Signature specification.
- Lines (023) to (027) provide a hint as to where to find the security token associated with this signature. Specifically, lines (024) to (025) indicate that the security token can be found at (pulled from) the specified URL.

UNIT-4

E-Commerce Infrastructure

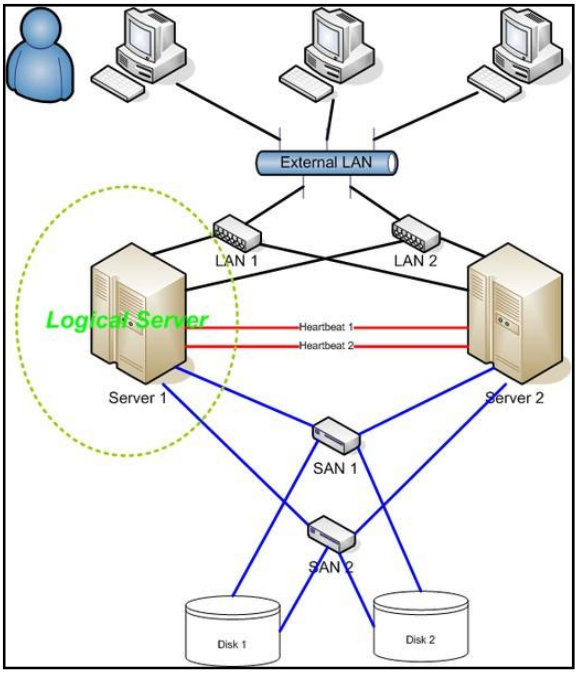
Cluster of Servers

A computer cluster is a group of linked computers, working together closely thus in many respects forming a single computer. The components of a cluster are commonly, but not always, connected to each other through fast local area networks.

Clusters are usually deployed to improve performance and/or availability over that of a single computer, while typically being much more cost-effective than single computers of comparable speed or availability.

Cluster Categorizations

- High-availability (HA) clusters
  - High-availability clusters (also known as Failover Clusters) are implemented primarily for the purpose of improving the availability of services that the cluster provides.
  - They operate by having redundant nodes, which are then used to provide service when system components fail.
  - The most common size for an HA cluster is two nodes, which is the minimum requirement to provide redundancy.
  - HA cluster implementations attempt to use redundancy of cluster components to eliminate single points of failure.
  - There are commercial implementations of High-Availability clusters for many operating systems.
  - One such implementation is the Gridlock platform from http://www.obsidiandynamics.com.



  - The Linux-HA project is one commonly used free software HA package for the Linux operating system.
  - The LanderCluster from Lander Software can run on Windows, Linux, and UNIX platforms.
- Load-balancing clusters
  - Load-balancing is when multiple computers are linked together to share computational workload or function as a single virtual computer.
  - Logically, from the user side, they are multiple machines, but function as a single virtual machine.
  - Requests initiated from the user are managed by, and distributed among, all the standalone computers to form a cluster.
  - This results in balanced computational work among different machines, improving the performance of the cluster systems.
- Compute Clusters
  - Often clusters are used primarily for computational purposes, rather than handling IO-oriented operations such as web service or databases.
  - For instance, a cluster might support computational simulations of weather or vehicle crashes.
  - The primary distinction within compute clusters is how tightly-coupled the individual nodes are.
  - For instance, a single compute job may require frequent communication among nodes - this implies that the cluster shares a dedicated network, is densely located, and probably has homogenous nodes.

    - This cluster design is usually referred to as Beowulf Cluster.
    - The other extreme is where a compute job uses one or few nodes, and needs little or no inter-node communication.
    - This latter category is sometimes called "Grid" computing.
    - Tightly-coupled compute clusters are designed for work that might traditionally have been called "supercomputing".
    - Middleware such as MPI (Message Passing Interface) or PVM (Parallel Virtual Machine) permits compute clustering programs to be portable to a wide variety of clusters.
- Grid Computing
    - Grids are usually computer clusters, but more focused on throughput like a computing utility rather than running fewer, tightly-coupled jobs.
    - Often, grids will incorporate heterogeneous collections of computers, possibly distributed geographically, sometimes administered by unrelated organizations.
    - Grid computing is optimized for workloads which consist of many independent jobs or packets of work, which do not have to share data between the jobs during the computation process.
    - Grids serve to manage the allocation of jobs to computers which will perform the work independently of the rest of the grid cluster.
    - Resources such as storage may be shared by all the nodes, but intermediate results of one job do not affect other jobs in progress on other nodes of the grid.
    - An example of a very large grid is the Folding@home project.
    - It is analyzing data that is used by researchers to find cures for diseases such as Alzheimer's and cancer.
    - Another large project is the SETI@home project, which may be the largest distributed grid in existence.
    - It uses approximately three million home computers all over the world to analyze data from the Arecibo Observatory radiotelescope, searching for evidence of extraterrestrial intelligence.
    - In both of these cases, there is no inter-node communication or shared storage.
    - Individual nodes connect to a main, central location to retrieve a small processing job.
    - They then perform the computation and return the result to the central server.
    - In the case of the @home projects, the software is generally run when the computer is otherwise idle.

A cluster of servers allows servers to work together as computer cluster, to provide failover and increased availability of applications, or parallel calculating power in case of high-performance computing (HPC) clusters (as in supercomputing).

A server cluster is a group of independent servers working together as a single system to provide high availability of services for clients. When a failure occurs on one computer in a cluster, resources are redirected and the workload is redistributed to another computer in the cluster.

You can use server clusters to ensure that users have constant access to important server-based resources. Server clusters are designed for applications that have long-running in-memory state or frequently updated data. Typical uses for server clusters include file servers, print servers, database servers, and messaging servers.

A cluster consists of two or more computers working together to provide a higher level of availability, reliability, and scalability than can be obtained by using a single computer. Microsoft cluster technologies guard against three specific types of failure:

- Application and service failures, which affect application software and essential services.

- System and hardware failures, which affect hardware components such as CPUs, drives, memory, network adapters, and power supplies.
- Site failures in multisite organizations, which can be caused by natural disasters, power outages, or connectivity outages.

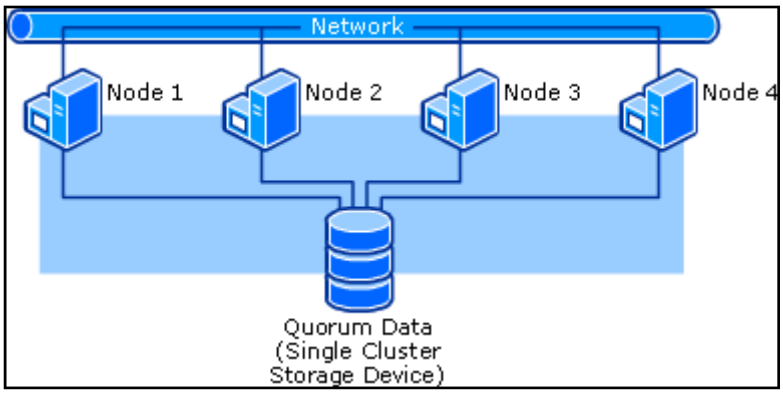
The ability to handle failure allows server clusters to meet requirements for high availability, which is the ability to provide users with access to a service for a high percentage of time while reducing unscheduled outages.

In a server cluster, each server owns and manages its local devices and has a copy of the operating system and the applications or services that the cluster is managing. Devices common to the cluster, such as disks in common disk arrays and the connection media for accessing those disks, are owned and managed by only one server at a time. For most server clusters, the application data is stored on disks in one of the common disk arrays, and this data is accessible only to the server that currently owns the corresponding application or service.

Server clusters are designed so that the servers in the cluster work together to protect data, keep applications and services running after failure on one of the servers, and maintain consistency of the cluster configuration over time.

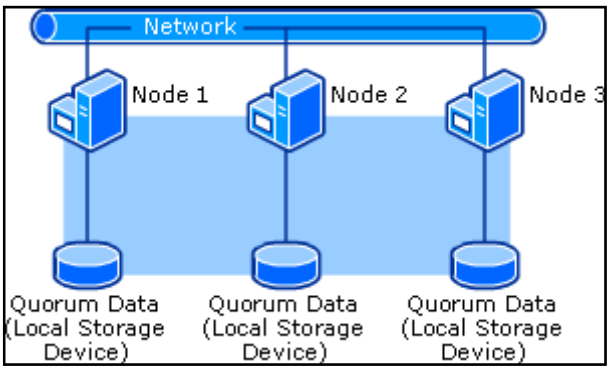
Types of Server Clusters

Single Quorum Device Cluster



The most widely used cluster type is the single quorum device cluster, also called the standard quorum cluster. In this type of cluster there are multiple nodes with one or more cluster disk arrays, also called the cluster storage, and a connection device, that is, a bus. Each disk in the array is owned and managed by only one server at a time. The disk array also contains the quorum resource. The following figure illustrates a single quorum device cluster with one cluster disk array. Because single quorum device clusters are the most widely used cluster, this Technical Reference focuses on this type of cluster.

Majority Node Set Cluster

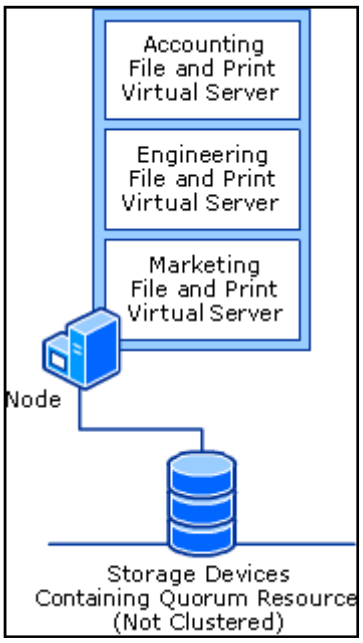


Windows Server 2003 supports another type of cluster, the majority node set cluster. In a majority node set cluster, each node maintains its own copy of the cluster configuration data. The quorum resource keeps configuration data consistent across the nodes. For this reason, majority node set clusters can be used for geographically dispersed clusters. Another advantage of majority node set clusters is that a quorum disk can be taken offline for maintenance and the cluster as a whole will continue to operate.

The major difference between majority node set clusters and single quorum device clusters is that single quorum device clusters can operate with just one node, but majority node set clusters need to have a majority of the cluster nodes available for the server cluster to operate. The following figure illustrates a majority node set cluster. For the cluster in the figure to continue to operate, two of the three cluster nodes (a majority) must be available.

Local Quorum Cluster

A local quorum cluster, also called a single node cluster, has a single node and is often used for testing. The following figure illustrates a local quorum cluster.



Virtualization

Server virtualization is the masking of server resources, including the number and identity of individual physical servers, processors, and operating systems, from server users. The server administrator uses a software application to divide one physical server into multiple isolated virtual environments. The virtual environments are sometimes called virtual private servers, but they are also known as guests, instances, containers or emulations.

Virtualization was invented more than thirty years ago to allow large expensive mainframes to be easily shared among different application environments. As hardware prices went down, the need for virtualization faded away. More recently, virtualization at all levels (system, storage, and network) became important again as a way to improve system security, reliability and availability, reduce costs, and provide greater flexibility.

Approaches to Virtualization

There are three popular approaches to server virtualization:

1. The virtual machine model,
2. The paravirtual machine model,
3. Virtualization at the operating system (OS) layer.

Virtual Machines

Virtual machines are based on the host/guest paradigm. Each guest runs on a virtual imitation of the hardware layer. This approach allows the guest operating system to run without modifications. It also allows the administrator to create guests that use different operating systems. The guest has no knowledge of the host's operating system because it is not aware that it's not running on real hardware.

It does, however, require real computing resources from the host -- so it uses a hypervisor to coordinate instructions to the CPU. The hypervisor is called a virtual machine monitor (VMM). It validates all the guest-issued CPU instructions and manages any executed code that requires addition privileges. VMware and Microsoft Virtual Server both use the virtual machine model.

Para-virtual Machine

The paravirtual machine (PVM) model is also based on the host/guest paradigm -- and it uses a virtual machine monitor too. In the paravirtual machine model, however, The VMM actually modifies the guest operating system's code. This modification is called porting. Porting supports the VMM so it can utilize privileged systems calls sparingly. Like virtual machines, paravirtual machines are capable of running multiple operating systems.

Virtualization @ OS Level

Virtualization at the OS level works a little differently. It isn't based on the host/guest paradigm. In the OS level model, the host runs a single OS kernel as its core and exports operating system functionality to each of the guests. Guests must use the same operating system as the host, although different distributions of the same system are allowed.

This distributed architecture eliminates system calls between layers, which reduces CPU usage overhead. It also requires that each partition remain strictly isolated from its neighbors so that a failure or security breach in one partition isn't able to affect any of the other partitions.

In this model, common binaries and libraries on the same physical machine can be shared, allowing an OS level virtual server to host thousands of guests at the same time.

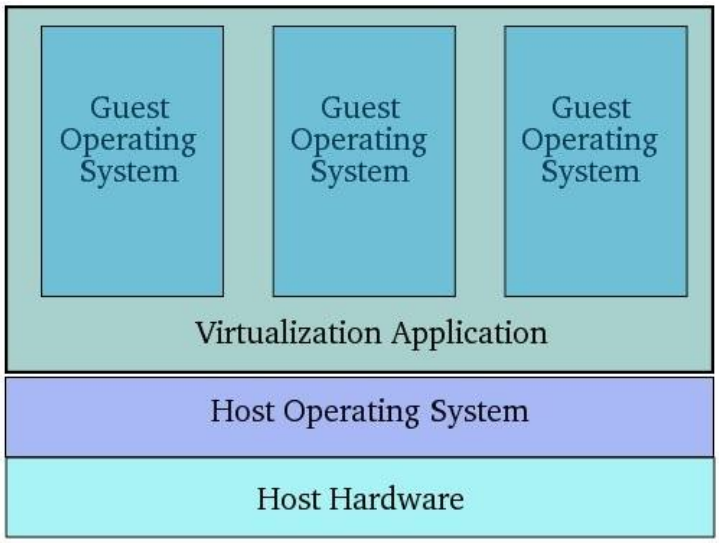
Virtualization Techniques

Guest Operating System Virtualization

In this scenario the physical host computer system runs a standard unmodified operating system such as Windows, Linux, Unix or MacOS X. Running on this operating system is a virtualization application which executes in much the same way as any other application such as a word processor or spreadsheet would run on the system. It is within this virtualization application that one or more virtual machines are created to run the guest operating systems on the host computer.

The virtualization application is responsible for starting, stopping and managing each virtual machine and essentially controlling access to physical hardware resources on behalf of the individual virtual machines. The virtualization application also engages in a process known as binary rewriting which involves scanning the instruction stream of the executing guest system and replacing any privileged instructions with safe emulations.

This has the effect of making the guest system think it is running directly on the system hardware, rather than in a virtual machine within an application.

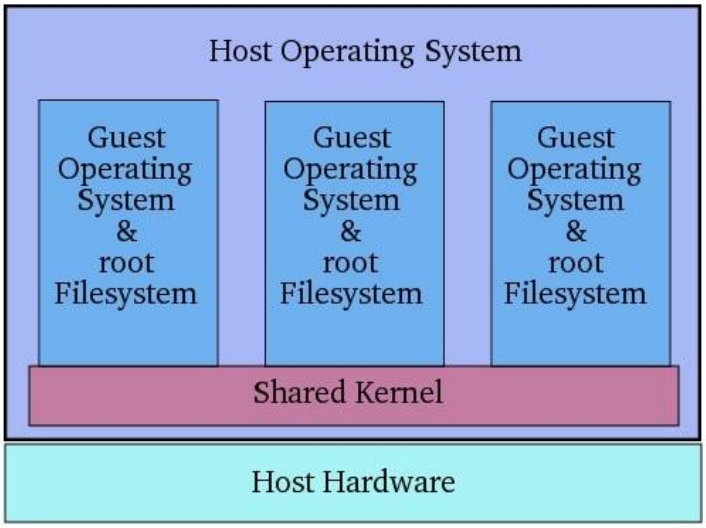


As outlined in the above diagram, the guest operating systems operate in virtual machines within the virtualization application which, in turn, runs on top of the host operating system in the same way as any other application. Clearly, the multiple layers of abstraction between the guest operating systems and the underlying host hardware are not conducive to high levels of virtual machine performance.

This technique does, however, have the advantage that no changes are necessary to either host or guest operating systems and no special CPU hardware virtualization support is required.

Shared Kernel Virtualization

Shared kernel virtualization (also known as system level or operating system virtualization) takes advantage of the architectural design of Linux and UNIX based operating systems. In order to understand how shared kernel virtualization works it helps to first understand the two main components of Linux or UNIX operating systems. At the core of the operating system is the kernel. The kernel, in simple terms, handles all the interactions between the operating system and the physical hardware. The second key component is the root file system which contains all the libraries, files and utilities necessary for the operating system to function. Under shared kernel virtualization the virtual guest systems each have their own root file system but share the kernel of the host operating system. This structure is illustrated in the following architectural diagram.

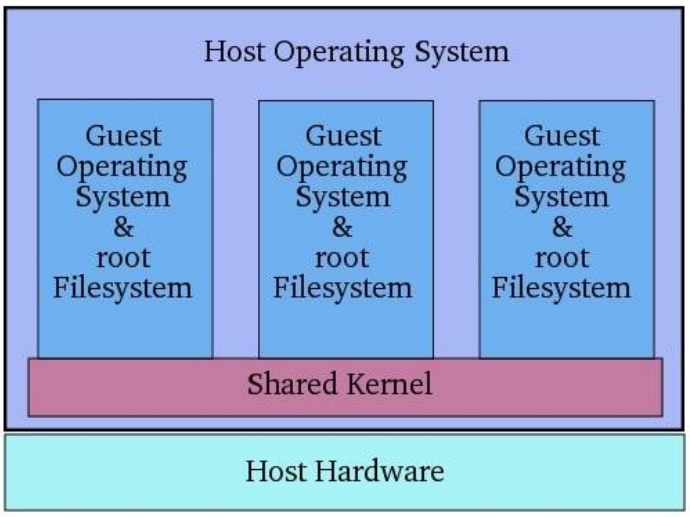


This type of virtualization is made possible by the ability of the kernel to dynamically change the current root file system (a concept known as chroot) to a different root file system without having to reboot the entire system. Essentially, shared kernel virtualization is an extension of this capability.

Perhaps the biggest single drawback of this form of virtualization is the fact that the guest operating systems must be compatible with the version of the kernel which is being shared. It is not, for example, possible to run Microsoft Windows as a guest on a Linux system using the shared kernel approach. Nor is it possible for a Linux guest system designed for the 2.6 version of the kernel to share a 2.4 version kernel.

Kernel Level Virtualization

Under kernel level virtualization the host operating system runs on a specially modified kernel which contains extensions designed to manage and control multiple virtual machines each containing a guest operating system. Unlike shared kernel virtualization each guest runs its own kernel, although similar restrictions apply in that the guest operating systems must have been compiled for the same hardware as the kernel in which they are running. Examples of kernel level virtualization technologies include User Mode Linux (UML) and Kernel-based Virtual Machine (KVM). The following diagram provides an overview of the kernel level virtualization architecture:



Cloud Computing

Cloud computing is Internet-based computing, whereby shared resources, software, and information are provided to computers and other devices on demand, like the electricity grid. Cloud computing is a paradigm shift following the shift from mainframe to client–server in the early 1980s. Details are abstracted from the users, who no longer have need for expertise in, or control over, the technology infrastructure "in the cloud" that supports them.

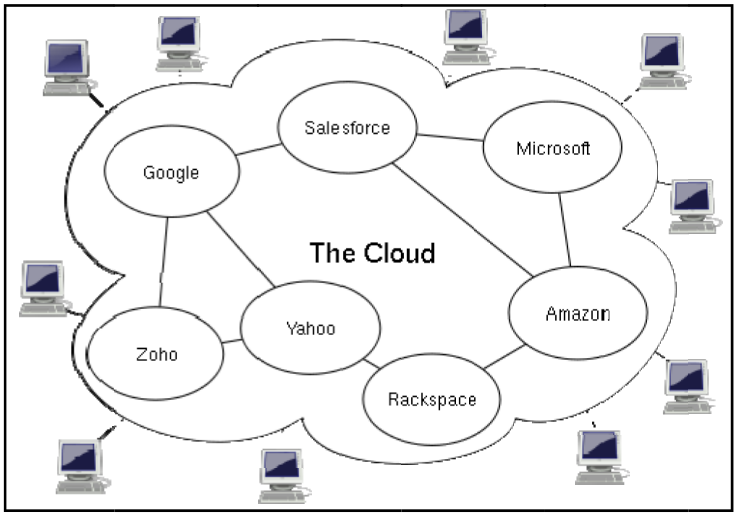
Cloud computing describes a new supplement, consumption, and delivery model for IT services based on the Internet, and it typically involves over-the-Internet provision of dynamically scalable and often virtualized resources. It is a byproduct and consequence of the ease-of-access to remote computing sites provided by the Internet. This frequently takes the form of web-based tools or applications that users can access and use through a web browser as if it were a program installed locally on their own computer.

NIST provides a somewhat more objective and specific definition here. The term "cloud" is used as a metaphor for the Internet, based on the cloud drawing used in the past to represent the telephone network and later to depict the Internet in computer network diagrams as an abstraction of the underlying infrastructure it represents.

Typical cloud computing providers deliver common business applications online that are accessed from another Web service or software like a Web browser, while the software and data are stored on servers. A key element of cloud computing is customization and the creation of a user-defined experience.

Most cloud computing infrastructures consist of services delivered through common centers and built on servers.

Clouds often appear as single points of access for consumers' computing needs. Commercial offerings are generally expected to meet quality of service (QoS) requirements of customers, and typically include service level agreements (SLAs).The major cloud service providers include Salesforce, Amazon and Google. Some of the larger IT firms that are actively involved in cloud computing are Microsoft, Hewlett Packard and IBM.



Cloud computing derives characteristics from, but should not be confused with:

1. Autonomic computing — "computer systems capable of self-management"
2. Client–server model – client–server computing refers broadly to any distributed application that distinguishes between service providers (servers) and service requesters (clients).
3. Grid computing — "a form of distributed computing and parallel computing, whereby a 'super and virtual computer' is composed of a cluster of networked, loosely coupled computers acting in concert to perform very large tasks"
4. Mainframe computer — powerful computers used mainly by large organizations for critical applications, typically bulk data-processing such as census, industry and consumer statistics, enterprise resource planning, and financial transaction processing.
5. Utility computing — the "packaging of computing resources, such as computation and storage, as a metered service similar to a traditional public utility, such as electricity";
6. Peer-to-peer – a distributed architecture without the need for central coordination, with participants being at the same time both suppliers and consumers of resources (in contrast to the traditional client–server model)

Characteristics

In general, cloud computing customers do not own the physical infrastructure, instead avoiding capital expenditure by renting usage from a third-party provider. They consume resources as a service and pay only for resources that they use. Many cloud-computing offerings employ the utility computing model, which is analogous to how traditional utility services (such as electricity) are consumed, whereas others bill on a subscription basis.

Sharing "perishable and intangible" computing power among multiple tenants can improve utilization rates, as servers are not unnecessarily left idle (which can reduce costs significantly while increasing the speed of application development).

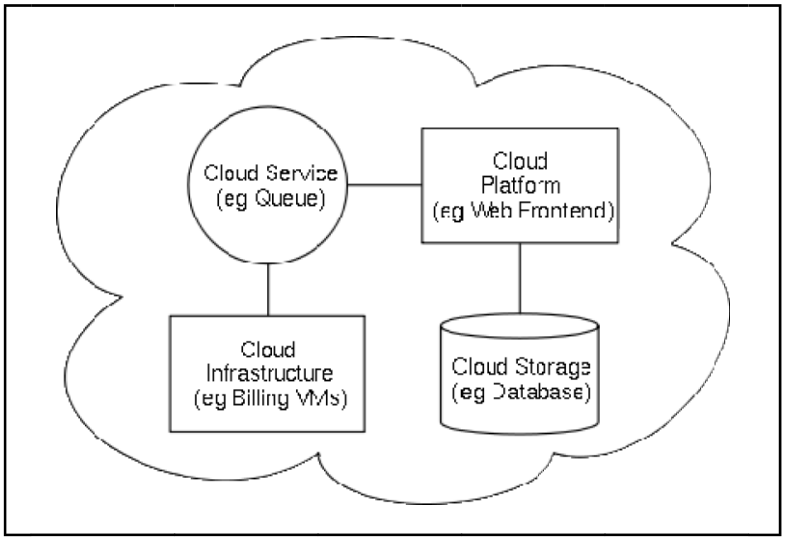
A side-effect of this approach is that overall computer usage rises dramatically, as customers do not have to engineer for peak load limits. In addition, "increased high-speed bandwidth" makes it possible to receive the same. The cloud is becoming increasingly associated with small and medium enterprises (SMEs) as in many cases they cannot justify or afford the large capital expenditure of traditional IT.

SMEs also typically have less existing infrastructure, less bureaucracy, more flexibility, and smaller capital budgets for purchasing in-house technology. Similarly, SMEs in emerging markets are typically unburdened by established legacy infrastructures, thus reducing the complexity of deploying cloud solutions.

Architecture

Cloud architecture, the systems architecture of the software systems involved in the delivery of cloud computing, typically involves multiple cloud components communicating with each other over applicati n programming interfaces, usually web services.

This resembles the Unix philosophy of having multiple programs each doing one thing well and working together over universal interfaces. Complexity is controlled and the resulting systems are more manageable than their monolithic counterparts.



The two most significant components of cloud computing architecture are known as the front end and the back end. The front end is the part seen by the client, i.e. the computer user. This includes the client's network (or computer) and the applications used to access the cloud via a user interface such as a web browser.

The back end of the cloud computing architecture is the 'cloud' itself, comprising various computers, servers and data storage devices.

Key Features

- **Agility** improves with users' ability to rapidly and inexpensively re-provision technological infrastructure resources.
- **Cost** is claimed to be greatly reduced and capital expenditure is converted to operational expenditure. This ostensibly lowers barriers to entry, as infrastructure is typically provided by a third-party and does not need to be purchased for one-time or infrequent intensive computing tasks. Pricing on a utility computing basis is fine-grained with usage-based options and fewer IT skills are required for implementation (in-house).

- **Device and location independence** enable users to access systems using a web browser regardless of their location or what device they are using (e.g., PC, mobile). As infrastructure is off-site (typically provided by a third-party) and accessed via the Internet, users can connect from anywhere.
- **Multi-tenancy** enables sharing of resources and costs across a large pool of users thus allowing for:
    - Centralization of infrastructure in locations with lower costs (such as real estate, electricity, etc.)
    - Peak-load capacity increases (users need not engineer for highest possible load-levels)
    - Utilization and efficiency improvements for systems that are often only 10–20% utilized.
- **Reliability** is improved if multiple redundant sites are used, which makes well designed cloud computing suitable for business continuity and disaster recovery.[38] Nonetheless, many major cloud computing services have suffered outages, and IT and business managers can at times do little when they are affected.
- **Scalability** via dynamic ("on-demand") provisioning of resources on a fine-grained, self-service basis near real-time, without users having to engineer for peak loads. Performance is monitored, and consistent and loosely coupled architectures are constructed using web services as the system interface. One of the most important new methods for overcoming performance bottlenecks for a large class of applications is data parallel programming on a distributed data grid.
- **Security** could improve due to centralization of data, increased security-focused resources, etc., but concerns can persist about loss of control over certain sensitive data, and the lack of security for stored kernels. Security is often as good as or better than under traditional systems, in part because providers are able to devote resources to solving security issues that many customers cannot afford. Providers typically log accesses, but accessing the audit logs themselves can be difficult or impossible. Furthermore, the complexity of security is greatly increased when data is distributed over a wider area and / or number of devices.
- **Maintenance** of cloud computing applications is easier, since they don't have to be installed on each user's computer. They are easier to support and to improve since the changes reach the clients instantly.
- **Metering** means that cloud computing resources usage should be measurable and should be metered per client and application on a daily, weekly, monthly, and yearly basis.

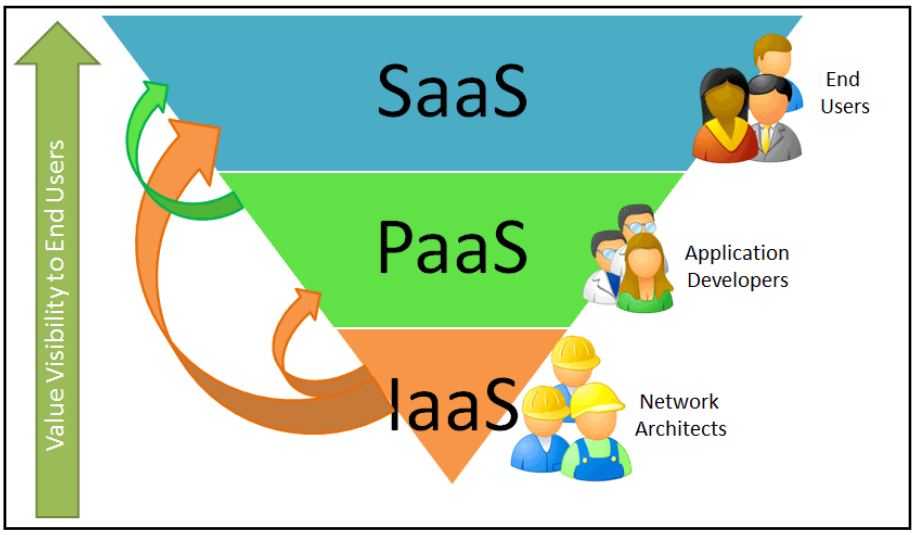
Cloud computing provides the means through which everything — from computing power to computing infrastructure, applications, business processes to personal collaboration — can be delivered to you as a service wherever and whenever you need.

Deployment Models

Cloud computing is offered in different forms:

- **Public clouds**
    - Public cloud or external cloud describes cloud computing in the traditional mainstream sense, whereby resources are dynamically provisioned on a fine-grained, self-service basis over the Internet, via web applications/web services, from an off-site third-party provider who bills on a fine-grained utility computing basis.
- **Community Clouds**
    - A community cloud may be established where several organizations have similar requirements and seek to share infrastructure so as to realize some of the benefits of cloud computing. With the costs spread over fewer users than a public cloud (but more than a single tenant) this option is more expensive but may offer a higher level of privacy, security and/or policy compliance.
- Private clouds
- Hybrid clouds
    - A hybrid cloud environment consisting of multiple internal and/or external providers "will be typical for most enterprises". By integrating multiple cloud services users may be able to ease the transition to public cloud services.

Cloud Computing Layers



Infrastructure As A Service (IAAS)

IaaS is the delivery of computer hardware (servers, networking technology, storage, and data centre space) as a service. You also can expect it to include the delivery of operating systems and virtualization technology to manage the resources. The IaaS customers rent computing resources instead of buying and installing them in their own data centre. Consumers control and manage the systems in terms of the operating systems, applications, storage, and network connectivity, but do not themselves control the cloud infrastructure.

- Much of the IaaS market will likely follow in the path of the ISP market. The ISP model has been proven and some large ISPs (GoDaddy at www.godaddy. com and inMotion hosting at www.inmotionhosting.com) run millions of Web sites.
- The ISP service is typically paid for based on the amount of resources used over time. This can include dynamic scaling so that if more resources are required than expected, they will be provided immediately (up to a given limit).
- The arrangement involves an agreed-upon service level — normally 99.9 percent availability or better, with limits set on CPU usage, memory, disk space, and Internet bandwidth. No one will object if you want to rent a server or a virtual server from an ISP and you run a data mart (instead of running a Web site).
- Nothing in the customer agreement stops you from using the resources in that way. It wouldn't make much sense, however, because you probably wouldn't get the service level agreement you wanted or the support you needed. IaaS takes the ISP model to a new level.

Platform As A Service (PAAS)

PaaS has many definitions, we'd like you to think about it as a computing platform that includes a set of development, middleware, and deployment capabilities. A key vendor characteristic is creating and encouraging a deep ecosystem of partners who all commit to this environment for the future.

When organizations are looking for capacity on demand, they often look to Infrastructure as a Service (IaaS). However, when an organization is looking for a deeper set of capabilities, they look at Platform as a Service (PaaS).

- PaaS has to leverage the Internet.
- PaaS must offer some type of development language so professional developers (and in some cases users) can add value.

- These environments need a way to monitor and measure resource use and to track overall performance of the vendor's platform.
- Almost all PaaS platforms are based on a multi-tenancy architecture (which lets multiple clients run their copy separately from each other through virtualization) so that each customer's code or data is isolated from others.
- A PaaS environment needs to support the development lifecycle and the team development process, including testing.
- A PaaS platform needs to include services interfaces such as SOAP (Simple Object Access Protocol) and XML (eXtensible Markup Language), among others.
- A PaaS platform must be able to deploy, manage, test, and maintain the developed applications.
- A PaaS platform must support well-defined and well-documented interfaces so elements and components can be used in the following:
    - Composite applications are created by combining services to create an enterprise application based on orchestration of business logic and rules.
    - Portals, which are an organized environment that organizes application components for the customer.
    - Mashups, which let end users easily bring together two or more business services that can communicate and exchange data.

Software As A Service (SAAS)

Mainframe systems were simply too expensive for most companies to buy their own systems. A couple of decades later, minicomputers, servers, and personal computers changed the dynamics of the market. Economically, it was feasible for any Tom, Dick, and Harriet to own their own systems and the software. Not all software moved to an internal model however. (Software such as ADP's payroll system, for example, remained Software as a Service.)

Two key events converged to create the model that we now call Software as a Service (SaaS):

- First, the Internet became a commercial platform.
- Second, software costs and complexities became so difficult that running, upgrading, and managing software become too complex for many companies to manage.
    - This was especially true for small- and medium sized companies that didn't want the expenses of managing all the components. These companies were the first to embrace this new generation of SaaS.

Characterizing Software as a Service:

What characteristics have to be in place for a SaaS to be commercially viable? Here's what we think is necessary:

- **The SaaS application needs to be generalized enough so that lots of customers will be interested in the service.** Here are some examples of these types of applications: accounting, collaboration, project management, testing, analytics, content management, Internet marketing, risk management and of course, CRM. What doesn't work as SaaS? A specialized one of- a-kind application with a small number of potential customers.
- **SaaS applications need sophisticated navigation and ease of use.** If an SaaS application isn't easy to use, customers will simply stop subscribing. Most SaaS vendors offer prospective customers a free trial for a month or so. If the customer doesn't start using the application during that first month, it's likely that the customer won't sign a contract. This is really important because it has been reported that less than 20 percent of users remain customers after the first month or so.
- **The SaaS application needs be modular and service oriented.** Without this modular approach, it will be hard to change and difficult to have third-party independent companies join the ecosystem.
- **An SaaS application needs to include measuring and monitoring so customers can be charged actual usage.**

- **An SaaS application must have a built-in billing service.**
- **SaaS applications need published interfaces and an ecosystem of partners who can expand the company's customer base and market reach.**
- **SaaS applications have to ensure that each customer's data and specialized configurations are separate and secure from other customers' data and configurations.**
- **SaaS applications need to provide sophisticated business process configurators for customers.** Each customer can change the process within the standardized SaaS application. For example, a company might want to add a process so a manager has to approve the price being offered to a new customer. A built-in configuration tool enables this to be done on an ad hoc basis without programming.
- **SaaS applications need to constantly provide fast releases of new features and new capabilities.** This must be done without impacting the customer's ability to continue business as usual.
- **SaaS applications have to protect the integrity of customer data.**
- That includes providing techniques for allowing data to migrate either to a private database inside the firewall or to a third-party storage capability.

Cloud Market Types	Types of Offerings	Examples
Software-as-a-Service	• Rich Internet application web sites • Application as Web Sites • Collaboration and email • Office Productivity • Client apps that connect to services in the cloud	• Flikr • Myspace.com • Cisco WebEx office • Gmail • IBM Bluehouse
App-components -as-a-Service	• APIs for specific service access for integration • Web-based software service than can combine to create new services, as in a mashup	• Amazon Flexible Payments Service and DevPay • Salesforce.com's AppExchange • Yahoo! Maps API • Google Calendar API • zembly
Platform-as-a-Service	• Development-platform-as-a-service • Database • Message Queue • App Servicer • Blob or object data stores	• Google App Engine and BigTable • Microsoft SQL Server Data Services • Engine Yard • Salesforce.com's Force.com
Infrastructure-as-a-Service	• Virtual servers • Logical disks • VLAN networks • Systems Management	• Akamai • Amazon EC2 • CohesiveFT • Mosso (from Rackspace) • Joyent Accelerators • Nirvanix Storage Delivery Network
Physical Infrastructure	• Managed Hosting • Collocation • Internet Service Provider • Unmanaged hosting	• GoDaddy.com • Rackspace • Savvis

Adapted from Forrester Research Taxonomy

Server Consolidation using Cloud

Server consolidation is an approach to the efficient usage of computer server resources in order to reduce the total number of servers or server locations that an organization requires. The practice developed in response to the problem of server sprawl, a situation in which multiple, under-utilized servers take up more space and consume more resources than can be justified by their workload.

- Servers in many companies typically run at 15-20% of their capacity,
    - which may not be a sustainable ratio in the current economic environment.
- Businesses are increasingly turning to server consolidation as one means of cutting unnecessary costs and maximizing return on investment (ROI) in the data center.
    - Of 518 respondents in a Gartner Group research study, six percent had conducted a server consolidation project,
    - 61% were currently conducting one, and
    - 28% were planning to do so in the immediate future.

Although consolidation can substantially increase the efficient use of server resources, it may also result in complex configurations of data, applications, and servers that can be confusing for the average user to contend with. To alleviate this problem, server virtualization may be used to mask the details of server resources from users while optimizing resource sharing. Another approach to server consolidation is the use of blade servers to maximize the efficient use of space.

There are four things to consider when looking at server consolidation: hardware, redundancy, operating system, and maximizing efficiency.

Cloud computing can be used for Server Consolidation by moving applications such as Exchange Mail, Data Storage etc. to a public cloud.

Note: Relate all the cloud layers and deployment models to server consolidation and explain the applications that can be housed remotely on a cloud to minimize the number, redundancy and underutilization of servers.

For many companies, the concept of server consolidation connotes sweeping projects that replace numerous smaller servers with powerful, expensive workhorses. Many organizations have pursued this approach and realized significant cost savings and improved efficiency. Achieving the benefits of server consolidation, however, does not necessarily require a large-scale effort that is both extensive and expensive in terms of a lengthy planning cycle and up-front investment. Many organizations consolidate as part of the natural refresh cycle of their technology infrastructures. These efforts are more iterative processes, rather than one-time projects, that focus on reducing the number of datacenters as the technology infrastructure is updated and augmented. The results are still significant. By consolidating, organizations are able to boost efficiency and improve their service capabilities while maintaining an infrastructure that is both robust, highly available, and adaptive to changing business requirements. Dell is introducing an expanded program aimed at consolidating Microsoft-based environments to help customers reduce their total costs and gain greater control over their IT infrastructure. The company hopes to use its new Server Consolidation ROI Analyst Tool, latest-generation enterprise servers, systems management software, and services to help organizations realize the benefits of consolidation as they renew their technology infrastructures.

Drivers & Benefits of Server Consolidation

A good deal of the impetus for consolidation stems from the proliferation of distributed systems that occurred during the 1980s and 1990s. As the price/performance and functionality of small and midrange systems improved dramatically, many organizations moved toward a model of deploying distributed servers to support departmental processes.

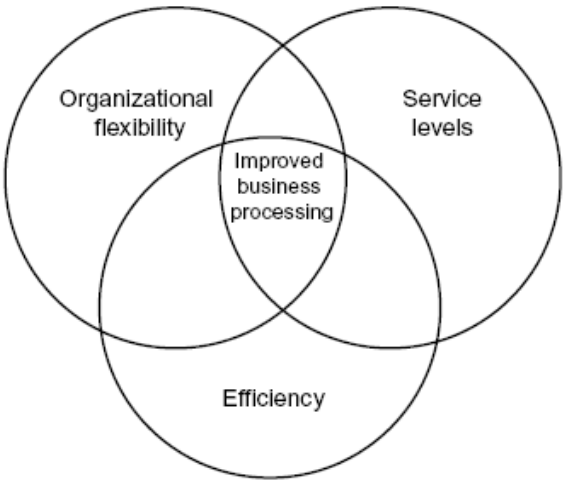
While such systems allowed organizations relatively cheap and easy access to powerful departmental applications, they eventually led to a quagmire of systems that were incompatible, unable to share information, and difficult or impossible to manage consistently. Organizations also faced the challenge of meeting increased levels of availability and security. At the same time, many companies looked to standardize their environments on a consolidated set of products from a more limited number of vendors in order to realize not only cost savings but also procurement, service, and management efficiencies.

At the same time, many IT organizations became more service oriented. Their focus shifted from managing processes and technology to providing information access to a broad set of user constituencies. Users began to expect universal access to applications and information.

Companies found that the duplication of servers, applications, and databases not only was expensive to maintain but also kept them from utilizing their resources and information to the maximum extent.

The response has been a trend toward recentralization and consolidation of IT resources in the datacenter. Consolidation allows companies to improve overall business processing through three primary IT objectives: a higher and more consistent level of service, greater efficiency and control over operations, and the flexibility to respond to constantly changing business requirements (see Figure 1).

Figure 1: Business Drivers for Consolidation



Source: IDC, 2002

ACHIEVING HIGHER LEVELS OF SERVICE

Providing high levels of service is on every IT manager's mind as the user base expands beyond internal users to customers, suppliers, students, patients, other government agencies, and business partners.

Everyone expects high levels of application and information availability and consistent response times. Consolidation provides a consistent management framework, which can lead to a more predictable and consistent level of service.

EFFICIENCY AND CONTROL

The key to gaining efficiency is in better utilizing the available IT resources. Consolidating the server environment can lead to a more disciplined approach to management. As much as 55% of IT costs are associated with personnel, and, in a consolidated environment, the productivity of administrative personnel increases greatly.

Some organizations may realize the benefit in reduced operational costs. IDC studies have shown a 7:1 cost savings in people management resources when processes and resources are consolidated. Others will benefit from the ability to focus highly skilled resources on higher-value tasks. Tighter management control will also enhance availability, security, and the ability to audit the consumption of services and charge back appropriately.

For IT managers who consistently cite floor space and power consumption as key issues, the benefits of consolidation are obvious. Many organizations are looking to reduce not only the number of servers but the number of datacenters. Because the cost of network bandwidth has dropped dramatically, companies no longer need to deploy servers as close as possible to the users they support. Therefore, many organizations can implement smaller, regional datacenters, instead of a datacenter at each location.

As server architectures grow increasingly modular, with the deployment of ultra-dense solutions such as server blades, the costs associated with power, space, cooling, and cabling will continue to drop.

GREATER ORGANIZATIONAL FLEXIBILITY

The greatest strategic value that companies gain from consolidation is an improved ability to efficiently adapt the infrastructure to incorporate new technologies and respond to new business requirements. All types of organizations — business, government, medical providers, and educational institutions — are driven by the need for high levels of services and information availability. With a consistent framework for managing data, IT spends less time moving data and transforming it into a usable form and, therefore, can respond more quickly to demands for data availability. The infrastructure is then better able to adapt as the organization moves forward. Once the IT center has completed the consolidation process and achieved the desired environment, users can expect more rapid deployment of new applications and features, leading to greater flexibility to respond to changing demands.

TYPES OF CONSOLIDATION

Consolidation of the server environment can occur at many different levels. Most organizations pursue different levels of consolidation at different times, depending on their particular requirements. Consolidation also tends to occur over time, as an iterative process, moving through the different types of consolidation.

The various types of consolidation should be viewed as steps along a continuum. The degrees of consolidation can be described in four general categories (see Figure 2):

→Logical consolidation reduces the number of points of control in the environment to a single administrative stream and reduces the number of consoles. The servers remain dispersed, while local operations are reduced or eliminated and management functions such as backup, restore, recovery, maintenance, and user support are performed remotely. A major benefit of logical consolidation is a reduction in operational headcount, or more efficient use of the skills already on hand. Using fewer people and a consistent set of products and processes can help reduce both the dilution of skills across the organization and the opportunity for errors. This approach reduces the cost of maintaining the environment and improves service to users.

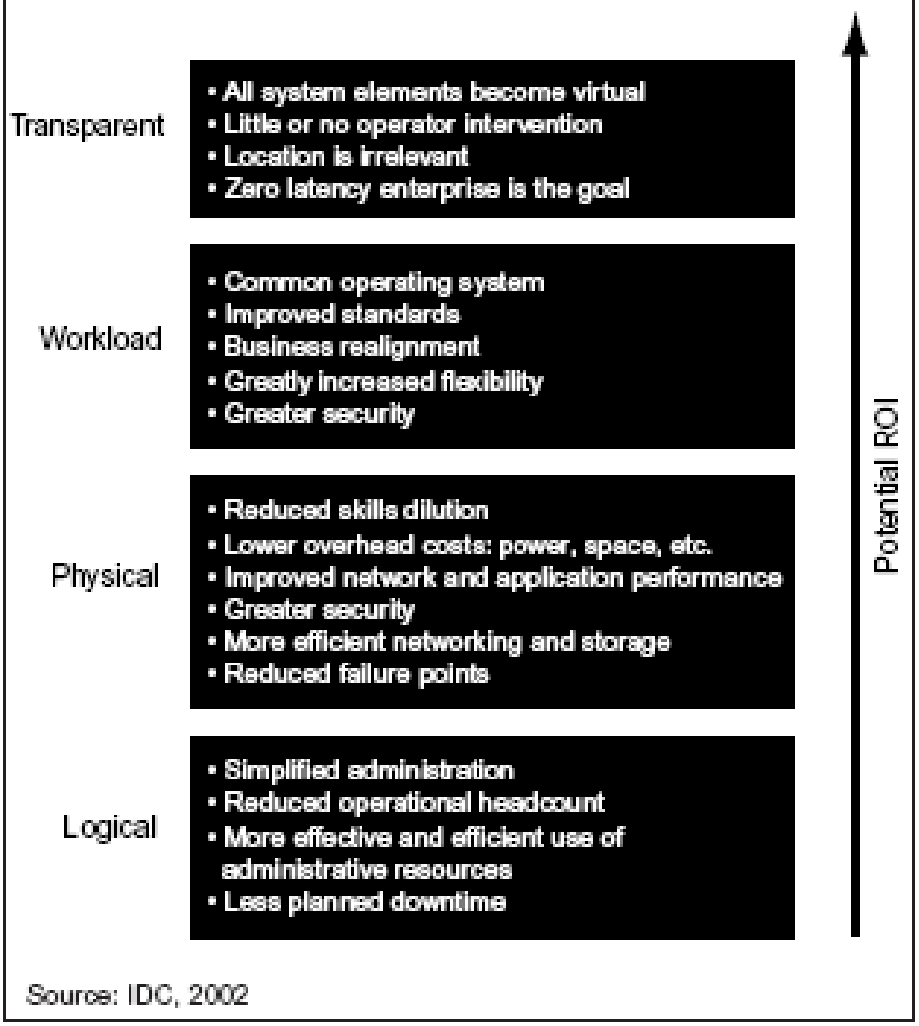
→Physical consolidation involves consolidating components of the IT environment in one physical location. This colocation leads to greater efficiency by eliminating the replication of skill sets across different locations. When systems are in a central location, networking becomes much easier and more efficient, power costs are reduced, backup can be performed more efficiently, and security can be increased. There are also subsequent savings in floor space costs. Consolidating locations also provides opportunity for configuring systems for higher availability.

→Workload consolidation reduces the number and variety of components in the environment. It involves not only servers but also other physical elements such as tapes, disks, network devices and connections, software, operating systems, and peripherals. The number of processes and procedures are also reduced. With fewer hardware and software standards to manage, IT departments can more easily move and change systems, applications, and peripherals. Network performance is enhanced even further, as are security and availability. Common operating

system environments allow more applications to share the same server, taking advantage of faster processors. Workload consolidation can be organized by application type, by operating system type, or by line of business.

→Transparent consolidation (including storage) involves pulling together a number of IT centers across a campus or network and implementing storage area networks to create a single set of resources. These environments will be highly automated, and their growth will be driven by the availability of high-speed, low-cost data networks.

Figure 2: Types of Consolidation and Their Associated Benefits



Source: IDC, 2002

Google Apps Engine

Google App Engine is a web application hosting service. By "web application," we mean an application or service accessed over the Web, usually with a web browser: storefronts with shopping carts, social networking sites, multiplayer games, mobile applications, survey applications, project management, collaboration, publishing, and all of the other things we're discovering are good uses for the Web. App Engine can serve traditional website content too, such as documents and images, but the environment is especially designed for real-time dynamic applications.

In particular, Google App Engine is designed to host applications with many simultaneous users. When an application can serve many simultaneous users without degrading performance, we say it scales. Applications written for App Engine scale automatically. As more people use the application, App Engine allocates more resources for the application and manages the use of those resources. The application itself does not need to know anything about the resources it is using.

Unlike traditional web hosting or self-managed servers, with Google App Engine, you only pay for the resources you use. These resources are measured down to the gigabyte, with no monthly fees or up-front charges. Billed resources include CPU usage, storage per month, incoming and outgoing bandwidth, and several resources specific to App Engine services.

To help you get started, every developer gets a certain amount of resources for free, enough for small applications with low traffic. Google estimates that with the free resources, an app can accommodate about 5 million page views a month.

App Engine can be described as three parts: the **runtime environment**, the **datastore**, and the **scalable services**. In this chapter, we'll look at each of these parts at a high level.

The Runtime Environment

An App Engine application responds to web requests. A web request begins when a client, typically a user's web browser, contacts the application with an HTTP request, such as to fetch a web page at a URL. When App Engine receives the request, it identifies the application from the domain name of the address, either an .appspot.com subdomain (provided for free with every app) or a subdomain of a custom domain name you have registered and set up with Google Apps. App Engine selects a server from many possible servers to handle the request, making its selection based on which server is most likely to provide a fast response. It then calls the application with the content of the HTTP request, receives the response data from the application, and returns the response to the client.

From the application's perspective, the runtime environment springs into existence when the request handler begins, and disappears when it ends. App Engine provides at least two methods for storing data that persists between requests (discussed later), but these mechanisms live outside of the runtime environment. By not retaining state in the runtime environment between requests—or at least, by not expecting that state will be retained between requests—App Engine can distribute traffic among as many servers as it needs to give every request the same treatment, regardless of how much traffic it is handling at one time.

Application code cannot access the server on which it is running in the traditional sense. An application can read its own files from the filesystem, but it cannot write to files, and it cannot read files that belong to other applications. An application can see environment variables set by App Engine, but manipulations of these variables do not necessarily persist between requests. An application cannot access the networking facilities of the server hardware, though it can perform networking operations using services.

In short, each request lives in its own "sandbox." This allows App Engine to handle a request with the server that would, in its estimation, provide the fastest response. There is no way to guarantee that the same server hardware will handle two requests, even if the requests come from the same client and arrive relatively quickly. Sandboxing

also allows App Engine to run multiple applications on the same server without the behaviour of one application affecting another.

In addition to limiting access to the operating system, the runtime environment also limits the amount of clock time, CPU use, and memory a single request can take. App Engine keeps these limits flexible, and applies stricter limits to applications that use up more resources to protect shared resources from "runaway" applications. A request has up to 30 seconds to return a response to the client. While that may seem like a comfortably large amount for a web app, App Engine is optimized for applications that respond in less than a second.

Also, if an application uses many CPU cycles, App Engine may slow it down so the app isn't hogging the processor on a machine serving multiple apps. A CPU-intensive request handler may take more clock time to complete.

The Datastore

Most useful web applications need to store information during the handling of a request for retrieval during a later request. A typical arrangement for a small website involves a single database server for the entire site, and one or more web servers that connect to the database to store or retrieve data. Using a single central database server makes it easy to have one canonical representation of the data, so multiple users accessing multiple web servers all see the same and most recent information.

But a central server is difficult to scale once it reaches its capacity for simultaneous connections. By far the most popular kind of data storage system for web applications in the past decade has been the relational database, with tables of rows and columns arranged for space efficiency and concision, and with indexes and raw computing power for performing queries, especially "join" queries that can treat multiple related records as a queryable unit.

Other kinds of data storage systems include hierarchical datastores (file systems, XML databases) and object databases. Each kind of database has pros and cons, and which type is best suited for an application depends on the nature of the application's data and how it is accessed. And each kind of database has its own techniques for growing past the first server.

Google App Engine's database system most closely resembles an object database. It is not a join-query relational database, and if you come from the world of relational-database-backed web applications (as I did), this will probably require changing the way you think about your application's data. As with the runtime environment, the design of the App Engine datastore is an abstraction that allows App Engine to handle the details of distributing and scaling the application, so your code can focus on other things.

The Services

The datastore's relationship with the runtime environment is that of a service: the application uses an API to access a separate system that manages all of its own scaling needs separately from the runtime environment. Google App Engine includes several other self-scaling services useful for web applications.

The memory cache (or memcache) service is a short-term key-value storage service. Its main advantage over the datastore is that it is fast, much faster than the datastore for simple storage and retrieval. The memcache stores values in memory instead of on disk for faster access. It is distributed like the datastore, so every request sees the same set of keys and values. However, it is not persistent like the datastore: if a server goes down, such as during a power failure, memory is erased. It also has a more limited sense of atomicity and transactionality than the datastore. As the name implies, the memcache service is best used as a cache for the results of frequently performed queries or calculations.

The application checks for a cached value, and if the value isn't there, it performs the query or calculation and stores the value in the cache for future use. App Engine applications can access other web resources using the URL Fetch service.

The service makes HTTP requests to other servers on the Internet, such as to retrieve pages or interact with web services. Since remote servers can be slow to respond, the URL Fetch API supports fetching URLs in the background while a request handler does other things, but in all cases the fetch must start and finish within the request handler's lifetime. The application can also set a deadline, after which the call is cancelled if the remote host hasn't responded.

App Engine applications can send messages using the Mail service. Messages can be sent on behalf of the application or on behalf of the user who made the request that is sending the email (if the message is from the user). Many web applications use email to notify users, confirm user actions, and validate contact information.

An application can also receive email messages. If an app is configured to receive email, a message sent to the app's address is routed to the Mail service, which delivers the message to the app in the form of an HTTP request to a request handler.

App Engine applications can send and receive instant messages to and from chat services that support the XMPP protocol, including Google Talk. An app sends an XMPP chat message by calling the XMPP service. As with incoming email, when someone sends a message to the app's address, the XMPP service delivers it to the app by calling a request handler.

UNIT-5

HADOOP

Data**!**

We live in the data age. It's not easy to measure the total volume of data stored electronically, but an IDC estimate put the size of the "digital universe" at 0.18 zettabytes in 2006, and is forecasting a tenfold growth by 2011 to 1.8 zettabytes.* A zettabyte is 10^{21} bytes, or equivalently one thousand exabytes, one million petabytes, or one billion terabytes. That's roughly the same order of magnitude as one disk drive for every person in the world. This flood of data is coming from many sources. Consider the following:

- The New York Stock Exchange generates about one terabyte of new trade data per day.
- Facebook hosts approximately 10 billion photos, taking up one petabyte of storage.
- Ancestry.com, the genealogy site, stores around 2.5 petabytes of data.
- The Internet Archive stores around 2 petabytes of data, and is growing at a rate of 20 terabytes per month.
- The Large Hadron Collider near Geneva, Switzerland, will produce about 15 petabytes of data per year.

So there's a lot of data out there. But you are probably wondering how it affects you. Most of the data is locked up in the largest web properties (like search engines), or scientific or financial institutions, isn't it? Does the advent of "Big Data," as it is being called, affect smaller organizations or individuals?

Arguably it does. Take photos, for example. My wife's grandfather was an avid photographer, and took photographs throughout his adult life. His entire corpus of medium format, slide, and 35mm film, when scanned in at high-resolution, occupies around 10 gigabytes. Compare this to the digital photos that my family took last year, which take up about 5 gigabytes of space. My family is producing photographic data at 35 times the rate my wife's grandfather's did, and the rate is increasing every year as it becomes easier to take more and more photos.

More generally, the digital streams that individuals are producing are growing apace. Microsoft Research's MyLifeBits project gives a glimpse of archiving of personal information that may become commonplace in the near future. MyLifeBits was an experiment where an individual's interactions—phone calls, emails, documents—were captured electronically and stored for later access. The data gathered included a photo taken every minute, which resulted in an overall data volume of one gigabyte a month. When storage costs come down enough to make it feasible to store continuous audio and video, the data volume for a future MyLifeBits service will be many times that.

The trend is for every individual's data footprint to grow, but perhaps more importantly the amount of data generated by machines will be even greater than that generated by people. Machine logs, RFID readers, sensor networks, vehicle GPS traces, retail transactions—all of these contribute to the growing mountain of data.

The volume of data being made publicly available increases every year too. Organizations no longer have to merely manage their own data: success in the future will be dictated to a large extent by their ability to extract value from other organizations' data.

Initiatives such as Public Data Sets on Amazon Web Services, Infochimps.org, and theinfo.org exist to foster the "information commons," where data can be freely (or in the case of AWS, for a modest price) shared for anyone to download and analyze. Mashups between different information sources make for unexpected and hitherto unimaginable applications.

Take, for example, the Astrometry.net project, which watches the Astrometry group on Flickr for new photos of the night sky. It analyzes each image, and identifies which part of the sky it is from, and any interesting celestial bodies, such as stars or galaxies. Although it's still a new and experimental service, it shows the kind of things that are possible when data (in this case, tagged photographic images) is made available and used for something (image

analysis) that was not anticipated by the creator.

It has been said that "More data usually beats better algorithms," which is to say that for some problems (such as recommending movies or music based on past preferences), however fiendish your algorithms are, they can often be beaten simply by having more data (and a less sophisticated algorithm). The good news is that Big Data is here. The bad news is that we are struggling to store and analyze it.

Data Storage and Analysis

The problem is simple: while the storage capacities of hard drives have increased massively over the years, access speeds—the rate at which data can be read from drives— have not kept up. One typical drive from 1990 could store 1370 MB of data and had a transfer speed of 4.4 MB/s,§ so you could read all the data from a full drive in around five minutes. Almost 20 years later one terabyte drives are the norm, but the transfer speed is around 100 MB/s, so it takes more than two and a half hours to read all the data off the disk.

This is a long time to read all data on a single drive—and writing is even slower. The obvious way to reduce the time is to read from multiple disks at once. Imagine if we had 100 drives, each holding one hundredth of the data. Working in parallel, we could read the data in under two minutes.

Only using one hundredth of a disk may seem wasteful. But we can store one hundred datasets, each of which is one terabyte, and provide shared access to them. We can imagine that the users of such a system would be happy to share access in return for shorter analysis times, and, statistically, that their analysis jobs would be likely to be spread over time, so they wouldn't interfere with each other too much.

There's more to being able to read and write data in parallel to or from multiple disks, though.

The first problem to solve is hardware failure: as soon as you start using many pieces of hardware, the chance that one will fail is fairly high. A common way of avoiding data loss is through replication: redundant copies of the data are kept by the system so that in the event of failure, there is another copy available. This is how RAID works, for instance, although Hadoop's filesystem, the Hadoop Distributed Filesystem (HDFS), takes a slightly different approach, as you shall see later.

The second problem is that most analysis tasks need to be able to combine the data in some way; data read from one disk may need to be combined with the data from any of the other 99 disks. Various distributed systems allow data to be combined from multiple sources, but doing this correctly is notoriously challenging. MapReduce provides a programming model that abstracts the problem from disk reads and writes, transforming it into a computation over sets of keys and values. The important point for the present discussion is that there are two parts to the computation, the map and the reduce, and it's the interface between the two where the "mixing" occurs. Like HDFS, MapReduce has reliability built-in.

This, in a nutshell, is what Hadoop provides: a reliable shared storage and analysis system. The storage is provided by HDFS, and analysis by MapReduce. There are other parts to Hadoop, but these capabilities are its kernel.

Comparison with Other Systems

The approach taken by MapReduce may seem like a brute-force approach. The premise is that the entire dataset—or at least a good portion of it—is processed for each query. But this is its power. MapReduce is a *batch* query processor, and the ability to run an ad hoc query against your whole dataset and get the results in a reasonable time is transformative. It changes the way you think about data, and unlocks data that was previously archived on tape or disk. It gives people the opportunity to innovate with data. Questions that took too long to get answered before can now be answered, which in turn leads to new questions and new insights.

For example, Mailtrust, Rackspace's mail division, used Hadoop for processing email logs. One ad hoc query they wrote was to find the geographic distribution of their users. In their words:

This data was so useful that we've scheduled the MapReduce job to run monthly and we will be using this data to help us decide which Rackspace data centers to place new mail servers in as we growJ

By bringing several hundred gigabytes of data together and having the tools to analyze it, the Rackspace engineers were able to gain an understanding of the data that they otherwise would never have had, and, furthermore, they were able to use what they had learned to improve the service for their customers.

RDBMS

Why can't we use databases with lots of disks to do large-scale batch analysis? Why is MapReduce needed?

The answer to these questions comes from another trend in disk drives: seek time is improving more slowly than transfer rate. Seeking is the process of moving the disk's head to a particular place on the disk to read or write data. It characterizes the latency of a disk operation, whereas the transfer rate corresponds to a disk's bandwidth.

If the data access pattern is dominated by seeks, it will take longer to read or write large portions of the dataset than streaming through it, which operates at the transfer rate. On the other hand, for updating a small proportion of records in a database, a traditional B-Tree (the data structure used in relational databases, which is limited by the rate it can perform seeks) works well. For updating the majority of a database, a B-Tree is less efficient than MapReduce, which uses Sort/Merge to rebuild the database.

In many ways, MapReduce can be seen as a complement to an RDBMS. (The differences between the two systems are shown in Table 1-1.) MapReduce is a good fit for problems that need to analyze the whole dataset, in a batch fashion, particularly for ad hoc analysis. An RDBMS is good for point queries or updates, where the dataset has been indexed to deliver low-latency retrieval and update times of a relatively small amount of data. MapReduce suits applications where the data is written once, and read many times, whereas a relational database is good for datasets that are continually updated.

Table 1-1. RDBMS compared to MapReduce

	Traditional RDBMS	**MapReduce**
Data size	Gigabytes	Petabytes
Access	Interactive and batch	Batch
Updates	Read and write many times	Write once, read many times
Structure	Static schema	Dynamic schema
Integrity	High	Low
Scaling	Nonlinear	Linear

Another difference between MapReduce and an RDBMS is the amount of structure in the datasets that they operate on. *Structured data* is data that is organized into entities that have a defined format, such as XML documents or database tables that conform to a particular predefined schema. This is the realm of the RDBMS. *Semi-structured data,* on the other hand, is looser, and though there may be a schema, it is often ignored, so it may be used only as a guide to the structure of the data: for example, a spreadsheet, in which the structure is the grid of cells, although the cells themselves may hold any form of data. *Unstructured data* does not have any particular internal structure: for example, plain text or image data. MapReduce works well on unstructured or semi-structured data, since it is

designed to interpret the data at processing time. In other words, the input keys and values for MapReduce are not an intrinsic property of the data, but they are chosen by the person analyzing the data.

Relational data is often *normalized* to retain its integrity, and remove redundancy. Normalization poses problems for MapReduce, since it makes reading a record a nonlocal operation, and one of the central assumptions that MapReduce makes is that it is possible to perform (high-speed) streaming reads and writes.

A web server log is a good example of a set of records that is *not* normalized (for example, the client hostnames are specified in full each time, even though the same client may appear many times), and this is one reason that logfiles of all kinds are particularly well-suited to analysis with MapReduce.

MapReduce is a linearly scalable programming model. The programmer writes two functions—a map function and a reduce function—each of which defines a mapping from one set of key-value pairs to another. These functions are oblivious to the size of the data or the cluster that they are operating on, so they can be used unchanged for a small dataset and for a massive one. More importantly, if you double the size of the input data, a job will run twice as slow. But if you also double the size of the cluster, a job will run as fast as the original one. This is not generally true of SQL queries.

Over time, however, the differences between relational databases and MapReduce systems are likely to blur. Both as relational databases start incorporating some of the ideas from MapReduce (such as Aster Data's and Greenplum's databases), and, from the other direction, as higher-level query languages built on MapReduce (such as Pig and Hive) make MapReduce systems more approachable to traditional database programmers.*

Grid Computing

The High Performance Computing (HPC) and Grid Computing communities have been doing large-scale data processing for years, using such APIs as Message Passing Interface (MPI). Broadly, the approach in HPC is to distribute the work across a cluster of machines, which access a shared filesystem, hosted by a SAN. This works well for predominantly compute-intensive jobs, but becomes a problem when nodes need to access larger data volumes (hundreds of gigabytes, the point at which MapReduce really starts to shine), since the network bandwidth is the bottleneck, and compute nodes become idle.

MapReduce tries to colocate the data with the compute node, so data access is fast since it is local.* This feature, known as *data locality,* is at the heart of MapReduce and is the reason for its good performance. Recognizing that network bandwidth is the most precious resource in a data center environment (it is easy to saturate network links by copying data around), MapReduce implementations go to great lengths to preserve it by explicitly modelling network topology. Notice that this arrangement does not preclude high-CPU analyses in MapReduce.

MPI gives great control to the programmer, but requires that he or she explicitly handle the mechanics of the data flow, exposed via low-level C routines and constructs, such as sockets, as well as the higher-level algorithm for the analysis. MapReduce operates only at the higher level: the programmer thinks in terms of functions of key and value pairs, and the data flow is implicit.

Coordinating the processes in a large-scale distributed computation is a challenge. The hardest aspect is gracefully handling partial failure—when you don't know if a remote process has failed or not—and still making progress with the overall computation. MapReduce spares the programmer from having to think about failure, since the implementation detects failed map or reduce tasks and reschedules replacements on machines that are healthy. MapReduce is able to do this since it is a *shared-nothing* architecture, meaning that tasks have no dependence on one other. (This is a slight oversimplification, since the output from mappers is fed to the reducers, but this is under the control of the MapReduce system; in this case, it needs to take more care rerunning a failed reducer than rerunning a failed map, since it has to make sure it can retrieve the necessary map outputs, and if not, regenerate them by running the relevant maps again.) So from the programmer's point of view, the order in which the tasks run

doesn't matter. By contrast, MPI programs have to explicitly manage their own checkpointing and recovery, which gives more control to the programmer, but makes them more difficult to write.

MapReduce might sound like quite a restrictive programming model, and in a sense it is: you are limited to key and value types that are related in specified ways, and mappers and reducers run with very limited coordination between one another (the mappers pass keys and values to reducers). A natural question to ask is: can you do anything useful or nontrivial with it?

The answer is yes. MapReduce was invented by engineers at Google as a system for building production search indexes because they found themselves solving the same problem over and over again (and MapReduce was inspired by older ideas from the functional programming, distributed computing, and database communities), but it has since been used for many other applications in many other industries. It is pleasantly surprising to see the range of algorithms that can be expressed in MapReduce, from image analysis, to graph-based problems, to machine learning algorithms."" It can't solve every problem, of course, but it is a general data-processing tool.

Volunteer Computing

When people first hear about Hadoop and MapReduce, they often ask, "How is it different from SETI@home?" SETI, the Search for Extra-Terrestrial Intelligence, runs a project called SETI@home in which volunteers donate CPU time from their otherwise idle computers to analyze radio telescope data for signs of intelligent life outside earth. SETI@home is the most well-known of many *volunteer computing* projects; others include the Great Internet Mersenne Prime Search (to search for large prime numbers) and Folding@home (to understand protein folding, and how it relates to disease).

Volunteer computing projects work by breaking the problem they are trying to solve into chunks called *work units,* which are sent to computers around the world to be analyzed. For example, a SETI@home work unit is about 0.35 MB of radio telescope data, and takes hours or days to analyze on a typical home computer. When the analysis is completed, the results are sent back to the server, and the client gets another work unit. As a precaution to combat cheating, each work unit is sent to three different machines, and needs at least two results to agree to be accepted.

Although SETI@home may be superficially similar to MapReduce (breaking a problem into independent pieces to be worked on in parallel), there are some significant differences. The SETI@home problem is very CPU-intensive, which makes it suitable for running on hundreds of thousands of computers across the world,[1]" since the time to transfer the work unit is dwarfed by the time to run the computation on it. Volunteers are donating CPU cycles, not bandwidth.

MapReduce is designed to run jobs that last minutes or hours on trusted, dedicated hardware running in a single data center with very high aggregate bandwidth interconnects. By contrast, SETI@home runs a perpetual computation on untrusted machines on the Internet with highly variable connection speeds and no data locality.

A Brief History of Hadoop

Hadoop was created by Doug Cutting, the creator of Apache Lucene, the widely used text search library. Hadoop has its origins in Apache Nutch, an open source web search engine, itself a part of the Lucene project.

The Origin of the Name "Hadoop"

The name Hadoop is not an acronym; it's a made-up name. The project's creator, Doug Cutting, explains how the name came about:

> *The name my kid gave a stuffed yellow elephant. Short, relatively easy to spell and pronounce, meaningless, and not used elsewhere: those are my naming criteria. Kids are good at generating such. Googol is a kid's term.*

Subprojects and "contrib" modules in Hadoop also tend to have names that are unrelated to their function, often with an elephant or other animal theme ("Pig," for example). Smaller components are given more descriptive (and

therefore more mundane) names. This is a good principle, as it means you can generally work out what something does from its name. For example, the jobtracker5 keeps track of MapReduce jobs.

Building a web search engine from scratch was an ambitious goal, for not only is the software required to crawl and index websites complex to write, but it is also a challenge to run without a dedicated operations team, since there are so many moving parts. It's expensive too: Mike Cafarella and Doug Cutting estimated a system supporting a 1-billion-page index would cost around half a million dollars in hardware, with a monthly running cost of $30,000." Nevertheless, they believed it was a worthy goal, as it would open up and ultimately democratize search engine algorithms.

Nutch was started in 2002, and a working crawler and search system quickly emerged. However, they realized that their architecture wouldn't scale to the billions of pages on the Web. Help was at hand with the publication of a paper in 2003 that described the architecture of Google's distributed filesystem, called GFS, which was being used in production at Google.# GFS, or something like it, would solve their storage needs for the very large files generated as a part of the web crawl and indexing process. In particular, GFS would free up time being spent on administrative tasks such as managing storage nodes. In 2004, they set about writing an open source implementation, the Nutch Distributed Filesystem (NDFS).

In 2004, Google published the paper that introduced MapReduce to the world.* Early in 2005, the Nutch developers had a working MapReduce implementation in Nutch, and by the middle of that year all the major Nutch algorithms had been ported to run using MapReduce and NDFS.

NDFS and the MapReduce implementation in Nutch were applicable beyond the realm of search, and in February 2006 they moved out of Nutch to form an independent subproject of Lucene called Hadoop. At around the same time, Doug Cutting joined Yahoo!, which provided a dedicated team and the resources to turn Hadoop into a system that ran at web scale (see sidebar). This was demonstrated in February 2008 when Yahoo! announced that its production search index was being generated by a 10,000-core Hadoop cluster.[f]

In January 2008, Hadoop was made its own top-level project at Apache, confirming its success and its diverse, active community. By this time Hadoop was being used by many other companies besides Yahoo!, such as Last.fm, Facebook.

In one well-publicized feat, the *New York Times* used Amazon's EC2 compute cloud to crunch through four terabytes of scanned archives from the paper converting them to PDFs for the Web.t The processing took less than 24 hours to run using 100 machines, and the project probably wouldn't have been embarked on without the combination of Amazon's pay-by-the-hour model (which allowed the NYT to access a large number of machines for a short period), and Hadoop's easy-to-use parallel programming model.

In April 2008, Hadoop broke a world record to become the fastest system to sort a terabyte of data. Running on a 910-node cluster, Hadoop sorted one terabyte in 209 seconds (just under 3% minutes), beating the previous year's winner of 297 seconds (described in detail in "TeraByte Sort on Apache Hadoop" on page 461). In November of the same year, Google reported that its MapReduce implementation sorted one terabyte in 68 seconds.

Hadoop at Yahoo!

Building Internet-scale search engines requires huge amounts of data and therefore large numbers of machines to process it. Yahoo! Search consists of four primary components: the *Crawler,* which downloads pages from web servers; the *WebMap,* which builds a graph of the known Web; the *Indexer,* which builds a reverse index to the best pages; and the *Runtime,* which answers users' queries. The WebMap is a graph that consists of roughly 1 trillion (10^{12}) edges each representing a web link and 100 billion (10^{11}) nodes each representing distinct URLs. Creating and analyzing such a large graph requires a large number of computers running for many days.

In early 2005, the infrastructure for the WebMap, named *Dreadnaught,* needed to be redesigned to scale up to more nodes. Dreadnaught had successfully scaled from 20 to 600 nodes, but required a complete redesign to scale up further. Dreadnaught is similar to MapReduce in many ways, but provides more flexibility and less structure. In particular, each fragment in a Dreadnaught job can send output to each of the fragments in the next stage of the job, but the sort was all done in library code. In practice, most of the WebMap phases were pairs that corresponded to MapReduce. Therefore, the WebMap applications would not require extensive refactoring to fit into MapReduce.

Eric Baldeschwieler (Eric14) created a small team and we starting designing and prototyping a new framework written in C++ modeled after GFS and MapReduce to replace Dreadnaught. Although the immediate need was for a new framework for WebMap, it was clear that standardization of the batch platform across Yahoo! Search was critical and by making the framework general enough to support other users, we could better leverage investment in the new platform.

At the same time, we were watching Hadoop, which was part of Nutch, and its progress. In January 2006, Yahoo! hired Doug Cutting, and a month later we decided to abandon our prototype and adopt Hadoop. The advantage of Hadoop over our prototype and design was that it was already working with a real application (Nutch) on 20 nodes. That allowed us to bring up a research cluster two months later and start helping real customers use the new framework much sooner than we could have otherwise. Another advantage, of course, was that since Hadoop was already open source, it was easier (although far from easy!) to get permission from Yahoo!'s legal department to work in open source. So we set up a 200-node cluster for the researchers in early 2006 and put the WebMap conversion plans on hold while we supported and improved Hadoop for the research users.

The Apache Hadoop Project

Today, Hadoop is a collection of related subprojects that fall under the umbrella of infrastructure for distributed computing. These projects are hosted by the Apache Software Foundation, which provides support for a community of open source software projects. Although Hadoop is best known for MapReduce and its distributed filesystem (HDFS, renamed from NDFS), the other subprojects provide complementary services, or build on the core to add higher-level abstractions. The subprojects, and where they sit in the technology stack, are shown in Figure 1-1 and described briefly here:

Core

A set of components and interfaces for distributed filesystems and general I/O (serialization, Java RPC, persistent data structures).

Avro

A data serialization system for efficient, cross-language RPC, and persistent data storage. (At the time of this writing, Avro had been created only as a new subpro-ject, and no other Hadoop subprojects were using it yet.)

MapReduce

A distributed data processing model and execution environment that runs on large clusters of commodity machines.

HDFS

A distributed filesystem that runs on large clusters of commodity machines.

Pig

A data flow language and execution environment for exploring very large datasets. Pig runs on HDFS and MapReduce clusters.

HBase

A distributed, column-oriented database. HBase uses HDFS for its underlying storage, and supports both batch-style computations using MapReduce and point queries (random reads).

ZooKeeper

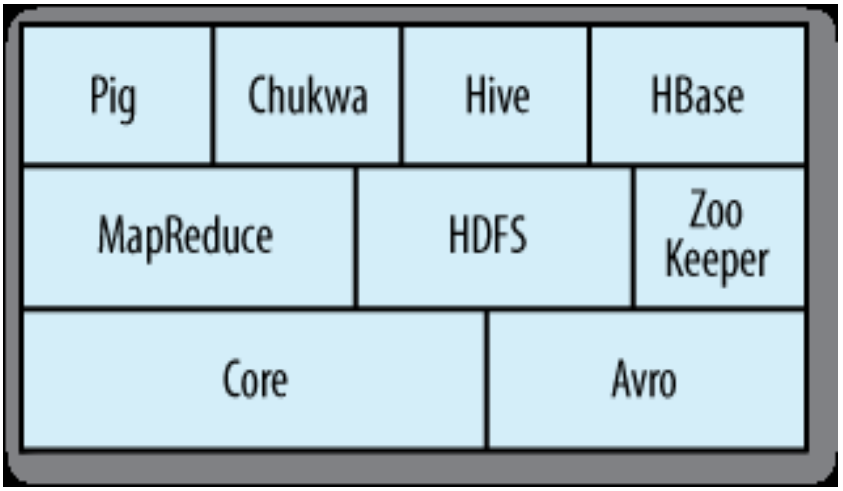
A distributed, highly available coordination service. ZooKeeper provides primitives such as distributed locks that can be used for building distributed applications.

Hive

A distributed data warehouse. Hive manages data stored in HDFS and provides a query language based on SQL (and which is translated by the runtime engine to MapReduce jobs) for querying the data.

Chukwa

A distributed data collection and analysis system. Chukwa runs collectors that store data in HDFS, and it uses MapReduce to produce reports. (At the time of this writing, Chukwa had only recently graduated from a "contrib" module in Core to its own subproject.).



HDFS

When a dataset outgrows the storage capacity of a single physical machine, it becomes necessary to partition it across a number of separate machines. Filesystems that manage the storage across a network of machines are called *distributed filesystems*. Since they are network-based, all the complications of network programming kick in, thus making distributed filesystems more complex than regular disk filesystems. For example, one of the biggest challenges is making the filesystem tolerate node failure without suffering data loss.

Hadoop comes with a distributed filesystem called HDFS, which stands for *Hadoop Distributed Filesystem.* (You may sometimes see references to "DFS"—informally or in older documentation or configuration—which is the same thing.) HDFS is Hadoop's flagship filesystem and is the focus of this chapter, but Hadoop actually has a general-purpose filesystem abstraction, so we'll see along the way how Hadoop integrates with other storage systems (such as the local filesystem and Amazon S3).

The Design of HDFS

HDFS is a filesystem designed for storing very large files with streaming data access patterns, running on clusters on commodity hardware. Let's examine this statement in more detail:

Very large files

"Very large" in this context means files that are hundreds of megabytes, gigabytes, or terabytes in size. There are Hadoop clusters running today that store petabytes of data.*

Streaming data access

HDFS is built around the idea that the most efficient data processing pattern is a write-once, read-many-times pattern. A dataset is typically generated or copied from source, then various analyses are performed on that dataset over time. Each analysis will involve a large proportion, if not all, of the dataset, so the time to read the whole dataset is more important than the latency in reading the first record.

Commodity hardware

Hadoop doesn't require expensive, highly reliable hardware to run on. It's designed to run on clusters of commodity hardware (commonly available hardware available from multiple vendors^) for which the chance of node failure across the cluster is high, at least for large clusters. HDFS is designed to carry on working without a noticeable interruption to the user in the face of such failure. It is also worth examining the applications for which using HDFS does not work so well. While this may change in the future, these are areas where HDFS is not a good fit today:

Low-latency data access

Applications that require low-latency access to data, in the tens of milliseconds range, will not work well with HDFS. Remember HDFS is optimized for delivering a high throughput of data, and this may be at the expense of latency. HBase is currently a better choice for low-latency access.

Lots of small files

Since the namenode holds filesystem metadata in memory, the limit to the number of files in a filesystem is governed by the amount of memory on the namenode. As a rule of thumb, each file, directory, and block takes

about 150 bytes. So, for example, if you had one million files, each taking one block, you would need at least 300 MB of memory. While storing millions of files is feasible, billions is beyond the capability of current hardware.

Multiple writers, arbitrary file modifications

Files in HDFS may be written to by a single writer. Writes are always made at the end of the file. There is no support for multiple writers, or for modifications at arbitrary offsets in the file. (These might be supported in the future, but they are likely to be relatively inefficient.)

HDFS Concepts Blocks

A disk has a block size, which is the minimum amount of data that it can read or write. Filesystems for a single disk build on this by dealing with data in blocks, which are an integral multiple of the disk block size. Filesystem blocks are typically a few kilobytes in size, while disk blocks are normally 512 bytes. This is generally transparent to the filesystem user who is simply reading or writing a file—of whatever length. However, there are tools to do with filesystem maintenance, such as *df* and *fsck,* that operate on the filesystem block level.

HDFS too has the concept of a block, but it is a much larger unit—64 MB by default. Like in a filesystem for a single disk, files in HDFS are broken into block-sized chunks, which are stored as independent units. Unlike a filesystem for a single disk, a file in HDFS that is smaller than a single block does not occupy a full block's worth of underlying storage. When unqualified, the term "block" in this book refers to a block in HDFS.

Why Is a Block in HDFS So Large?

HDFS blocks are large compared to disk blocks, and the reason is to minimize the cost of seeks. By making a block large enough, the time to transfer the data from the disk can be made to be significantly larger than the time to seek to the start of the block. Thus the time to transfer a large file made of multiple blocks operates at the disk transfer rate.

A quick calculation shows that if the seek time is around 10ms, and the transfer rate is 100 MB/s, then to make the seek time 1% of the transfer time, we need to make the block size around 100 MB. The default is actually 64 MB, although many HDFS installations use 128 MB blocks. This figure will continue to be revised upward as transfer speeds grow with new generations of disk drives.

This argument shouldn't be taken too far, however. Map tasks in MapReduce normally operate on one block at a time, so if you have too few tasks (fewer than nodes in the cluster), your jobs will run slower than they could otherwise.

Having a block abstraction for a distributed filesystem brings several benefits. The first benefit is the most obvious: a file can be larger than any single disk in the network. There's nothing that requires the blocks from a file to be stored on the same disk, so they can take advantage of any of the disks in the cluster. In fact, it would be possible, if unusual, to store a single file on an HDFS cluster whose blocks filled all the disks in the cluster.

Second, making the unit of abstraction a block rather than a file simplifies the storage subsystem. Simplicity is something to strive for all in all systems, but is important for a distributed system in which the failure modes are so varied. The storage subsystem deals with blocks, simplifying storage management (since blocks are a fixed size, it is easy to calculate how many can be stored on a given disk), and eliminating

metadata concerns (blocks are just a chunk of data to be stored—file metadata such as permissions information does not need to be stored with the blocks, so another system can handle metadata orthogonally).

Furthermore, blocks fit well with replication for providing fault tolerance and availability. To insure against corrupted blocks and disk and machine failure, each block is replicated to a small number of physically separate machines (typically three). If a block becomes unavailable, a copy can be read from another location in a way that is transparent to the client. A block that is no longer available due to corruption or machine failure can be replicated from their alternative locations to other live machines to bring the replication factor back to the normal level. (See "Data Integrity" on page 75 for more on guarding against corrupt data.) Similarly, some applications may choose to set a high replication factor for the blocks in a popular file to spread the read load on the cluster.

Like its disk filesystem cousin, HDFS's fsck command understands blocks. For example, running:

% hadoop fsck -files -blocks

will list the blocks that make up each file in the filesystem.

Namenodes and Datanodes

A HDFS cluster has two types of node operating in a master-worker pattern: a *name-node* (the master) and a number of *datanodes* (workers). The namenode manages the filesystem namespace. It maintains the filesystem tree and the metadata for all the files and directories in the tree. This information is stored persistently on the local disk in the form of two files: the namespace image and the edit log. The namenode also knows the datanodes on which all the blocks for a given file are located, however, it does not store block locations persistently, since this information is reconstructed from datanodes when the system starts.

A *client* accesses the filesystem on behalf of the user by communicating with the name-node and datanodes. The client presents a POSIX-like filesystem interface, so the user code does not need to know about the namenode and datanode to function.

Datanodes are the work horses of the filesystem. They store and retrieve blocks when they are told to (by clients or the namenode), and they report back to the namenode periodically with lists of blocks that they are storing.

Without the namenode, the filesystem cannot be used. In fact, if the machine running the namenode were obliterated, all the files on the filesystem would be lost since there would be no way of knowing how to reconstruct the files from the blocks on the datanodes. For this reason, it is important to make the namenode resilient to failure, and Hadoop provides two mechanisms for this.

The first way is to back up the files that make up the persistent state of the filesystem metadata. Hadoop can be configured so that the namenode writes its persistent state to multiple filesystems. These writes are synchronous and atomic. The usual configuration choice is to write to local disk as well as a remote NFS mount.

It is also possible to run a *secondary namenode,* which despite its name does not act as a namenode. Its main role is to periodically merge the namespace image with the edit log to prevent the edit log from becoming too large. The secondary namenode usually runs on a separate physical machine, since it requires plenty of CPU and as much memory as the namenode to perform the merge. It keeps a copy of the merged namespace image, which can be used in the event of the namenode failing. However, the state of the secondary namenode lags that of the primary, so in the event of total failure of the primary data, loss is almost guaranteed. The usual course of action in this case is to copy the namenode's metadata files that are on NFS to the secondary and run it as the new primary.

The Command-Line Interface

We're going to have a look at HDFS by interacting with it from the command line. There are many other interfaces to HDFS, but the command line is one of the simplest, and to many developers the most familiar.

We are going to run HDFS on one machine, so first follow the instructions for setting up Hadoop in pseudo-distributed mode in Appendix A. Later you'll see how to run on a cluster of machines to give us scalability and fault tolerance.

There are two properties that we set in the pseudo-distributed configuration that deserve further explanation. The first is fs.default.name, set to *hdfs://localhost/,* which is used to set a default filesystem for Hadoop. Filesystems are specified by a URI, and here we have used a hdfs URI to configure Hadoop to use HDFS by default. The HDFS daemons will use this property to determine the host and port for the HDFS namenode. We'll be running it on localhost, on the default HDFS port, 8020. And HDFS clients will use this property to work out where the namenode is running so they can connect to it.

We set the second property, dfs.replication, to one so that HDFS doesn't replicate filesystem blocks by the usual default of three. When running with a single datanode, HDFS can't replicate blocks to three datanodes, so it would perpetually warn about blocks being under-replicated. This setting solves that problem.

Basic Filesystem Operations

The filesystem is ready to be used, and we can do all of the usual filesystem operations such as reading files, creating directories, moving files, deleting data, and listing directories. You can type hadoop fs -help to get detailed help on every command. Start by copying a file from the local filesystem to HDFS:

% hadoop fs -copyFromLocal input/docs/quangle.txt hdfs://localhost/user/tom/quangle.txt

This command invokes Hadoop's filesystem shell command fs, which supports a number of subcommands—in this case, we are running -copyFromLocal. The local file *quangle.txt* is copied to the file */user/tom/quangle.txt* on the HDFS instance running on localhost. In fact, we could have omitted the scheme and host of the URI and picked up the default, hdfs://localhost, as specified in *core-site.xml.*

% **hadoop fs -copyFromLocal input/docs/quangle.txt /user/tom/quangle.txt**

We could also have used a relative path, and copied the file to our home directory in HDFS, which in this case is */user/tom:*

% **hadoop fs -copyFromLocal input/docs/quangle.txt quangle.txt**

Let's copy the file back to the local filesystem and check whether it's the same:

% **hadoop fs -copyToLocal quangle.txt quangle.copy.txt** % **md5 input/docs/quangle.txt quangle.copy.txt**

MD5 (input/docs/quangle.txt) = al6f23lda6b05e2ba7a339320e7dacd9 MD5 (quangle.copy.txt) = a16f231da6b05e2ba7a339320e7dacd9

The MD5 digests are the same, showing that the file survived its trip to HDFS and is back intact. Finally, let's look at an HDFS file listing. We create a directory first just to see how it is displayed in the listing:

% **hadoop fs -mkdir books** % **hadoop fs -ls .**

Found 2 items

drwxr-xr-x - tom supergroup 0 2009-04-02 22:41 /user/tom/books

-rw-r--r-- 1 tom supergroup 118 2009-04-02 22:29 /user/tom/quangle.txt

The information returned is very similar to the Unix command ls -l, with a few minor differences. The first column shows the file mode. The second column is the replication factor of the file (something a traditional Unix filesystems does not have). Remember we set the default replication factor in the site-wide configuration to be 1, which is why we see the same value here. The entry in this column is empty for directories since the concept of replication does not apply to them—directories are treated as metadata and stored by the namenode, not the datanodes. The third and fourth columns show the file owner and group. The fifth column is the size of the file in bytes, or zero for directories. The six and seventh columns are the last modified date and time. Finally, the eighth column is the absolute name of the file or directory.

File Permissions in HDFS

HDFS has a permissions model for files and directories that is much like POSIX. There are three types of permission: the read permission (r), the write permission (w) and the execute permission (x). The read permission is required to read files or list the contents of a directory.

The write permission is required to write a file, or for a directory, to create or delete files or directories in it.

The execute permission is ignored for a file since you can't execute a file on HDFS (unlike POSIX), and for a directory it is required to access its children.

Each file and directory has an *owner,* a *group,* and a *mode.* The mode is made up of the permissions for the user who is the owner, the permissions for the users who are members of the group, and the permissions for users who are neither the owner nor members of the group.

A client's identity is determined by the username and groups of the process it is running in. Because clients are remote, this makes it possible to become an arbitrary user, simply by creating an account of that name on the remote system.

Thus, permissions should be used only in a cooperative community of users, as a mechanism for sharing filesystem resources and for avoiding accidental data loss, and not for securing resources in a hostile environment. However, despite these drawbacks, it is worthwhile having permissions enabled (as it is by default; see the dfs.permissions property), to avoid accidental modification or deletion of substantial parts of the filesystem, either by users or by automated tools or programs.

When permissions checking is enabled, the owner permissions are checked if the client's username matches the owner, and the group permissions are checked if the client is a member of the group; otherwise, the other permissions are checked.

There is a concept of a super-user, which is the identity of the namenode process. Permissions checks are not performed for the super-user.

Hadoop Filesystems

Hadoop has an abstract notion of filesystem, of which HDFS is just one implementation. The Java abstract class org.apache.hadoop.fs. FileSystem represents a filesystem in Hadoop, and there are several concrete implementations, which are described below:

Filesystem	**URI scheme**	**Java implementation (all under org.apache.hadoop)**	**Description**
Local	*file*	fs.LocalFileSystem	A filesystem for a locally connected disk with client-side checksums. Use RawLocalFileSys tem for a local filesystem with no checksums. See "LocalFileSys-tem" on page 76.
HDFS	*hdfs*	hdfs.DistributedFileSystem	Hadoop's distributed filesystem. HDFS is designed to work efficiently in conjunction with Map-Reduce.
HFTP	*hftp*	hdfs.HftpFileSystem	A filesystem providing read-only access to HDFS over HTTP. (Despite its name, HFTP has no connection with FTP.) Often used with *distcp* ("Parallel Copying with distcp" on page 70) to copy data between HDFS clusters running different versions.
HSFTP	*hsftp*	hdfs.HsftpFileSystem	A filesystem providing read-only access to HDFS over HTTPS. (Again, this has no connection with FTP.)
HAR	*har*	fs.HarFileSystem	A filesystem layered on another filesystem for archiving files. Hadoop Archives are typically used for archiving files in HDFS to reduce the namenode's memory usage.
KFS (CloudStore)	*kfs*	fs.kfs.KosmosFileSystem	CloudStore (formerly Kosmos filesystem) is a distributed filesystem like HDFS or Google's GFS, written in C++. Find more information about it at *http://kosmosfs .sourceforge.net/*.
FTP	*ftp*	fs.ftp.FTPFileSystem	A filesystem backed by an FTP server.
S3 (native)	*s3n*	fs.s3native.NativeS3FileSystem	A filesystem backed by Amazon S3. See *http://wiki.apache.org/hadoop/AmazonS3*.
S3 (block-based)	*s3*	fs.s3.S3FileSystem	A filesystem backed by Amazon S3, which stores files in blocks

Note: Don’t go too much into the technical aspects of Hadoop & HDFS. You need to understand the concept and derive from that an understanding of unlocking old data. Fundamentally these technologies allow for mining data that otherwise could not be mined, simply by supporting distributed computing and storage. Also, use the slides to figure out how a MapReduce program works and the various entities involved.