JavaOne℠

Sun's 2004 Worldwide Java Developer Conference℠

High Performance Web services

Tackling Scalability & Speed

Ramesh Nagappan Staff Engineer

Sameer Tyagi Java Architect

Sun Microsystems

Java™

Sun microsystems

Goals For This Session

No Silver Bullet !

Take a detailed look at performance & scalability issues as well as mitigation strategies for real world Web Services

Agenda

Scalability & Performance

- Limiting factors for Web services
- Quantitative design and development
- Monitoring and measuring
- Problems & Mitigation Strategies
- Summary

Principles Critical in Production

- Scalability
  - Meet initial needs and grow rapidly
  - Respond to growth
    - Audience, Organization, Data
- Performance
  - Web Services should execute quickly
  - Complete the requested task quickly
  - Minimize delays in message delivery & processing times with increased traffic
- Predictability
  - Ensure predicable end to end latencies & response times
  - Comply with QoS requirements in SLA

Constraints in Production

- Complex multi-tier deployments
  - Inherent to the architecture
  - Increased number of hops and nodes
- Non deterministic transports
  - Heterogeneous execution environments
- Bit heavy content encoding
  - Text based rather than compact binary
- CPU intensive processing
  - SSL,XML parsing, XSLT, Header & Payload processing

Constraints in Production

- Capacity constraints
  - Software – Web, Application servers
  - Hardware - CPU, memory, bandwidth, shared infrastructures
  - Cost constraints- Setting up geographically dispersed mirror sites, with extra OC3s & an army of administrators, support on 24x7 standby - Can be a nightmare !
- Application design and configuration
  - Distributed, partitioned tiers
    - Presentation, business logic, integration, EIS
  - State maintenance
  - Cross domain, B2B
  - Integration with third party systems
- Internal processes,organizational structure & culture

Agenda

Scalability & Performance

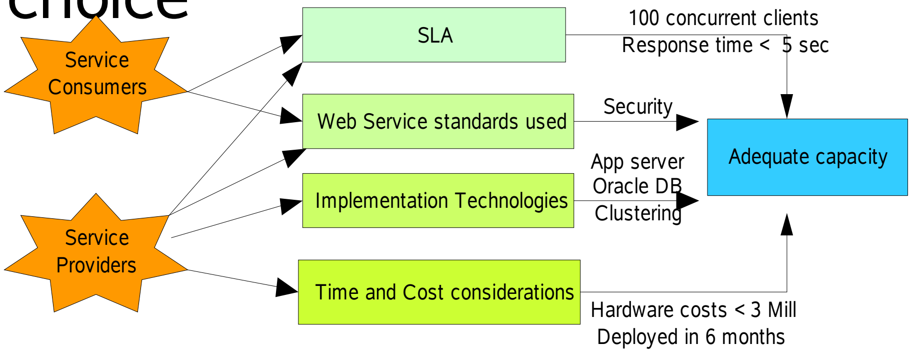
- Limiting factors for Web services
- Quantitative design and development
- Monitoring and measuring
- Problems & Mitigation Strategies
- Summary

What is Adequate Performance

- Web service performance can be analyzed from different view points
  - Service consumer: E.g. Reponses times, connection errors
  - Service producer: E.g. Transactions/sec, concurrent users
  - Process perspective: E.g.Time to perform business transaction
- Common metrics
  - End to end response time
  - Site response time
  - Throughput (requests/sec)
  - Throughput (Mbps)
  - HTTP or other errors /sec
  - Transactions per day

What is Adequate Capacity

- Adequate if
  - SLA is met, for specified standards, implementations and within costs
- Driven by Service Level Agreements (SLA)
  - Unlike web applications E.g. number of users known
  - SLA define tangible values to response times, availability, throughput
- Driven by standards
  - E.g. Digital signatures, SSL
- Driven by implementation choice
  - Containers, state, payload
- Driven by cost constraints
  - Budgets limit possible solutions

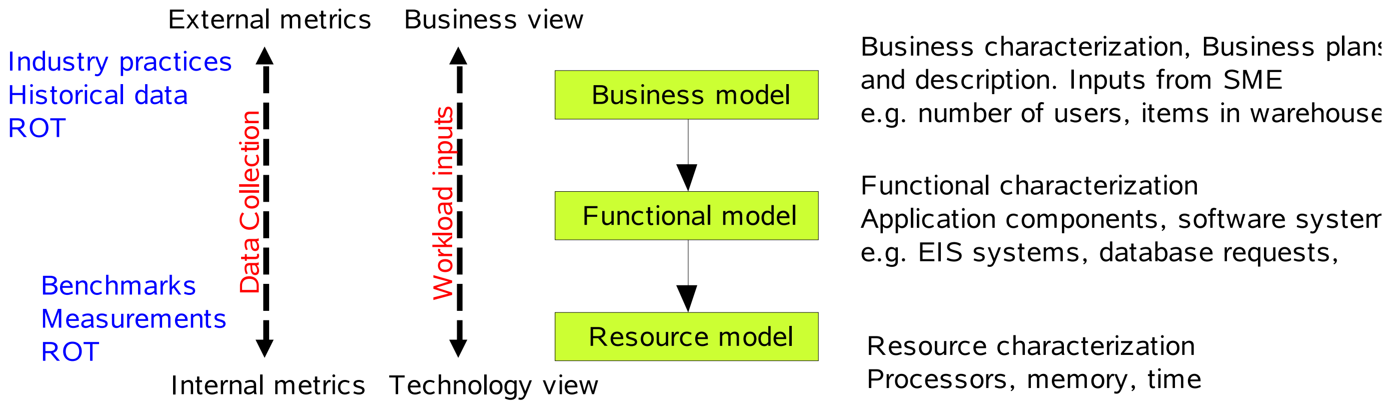


Modeling Capacity & Performance

- For Developers and Architects
  - During development
  - Validate the architecture, design & implementation
  - Predict when performance issues may pop up
  - Predict overall performance for components and Web services
  - Plan the capabilities of the Web services
- For IT support and maintenance
  - During operation
  - Developing contingency plans
  - Predict extensibility, scalability of the Web service
  - Establish ramp-up thresholds to provide for new and existing customers

Modeling Capacity

- Describe the workload for Web services
  - Captures resource demands
  - Workload parameters
    - Intensity : Requests/day, messages per day per customer, transaction rates, concurrent users etc
    - Service demand : Message size, number of users, CPU/Memory utilization etc
  - Uses a representative timeframe
  - Executable in nature (load tools, benchmarks, drivers)



External metrics

Business view

Industry practices
Historical data
ROT

Data Collection

Workload inputs

Business model

Functional model

Resource model

Benchmarks
Measurements
ROT

Internal metrics

Technology view

Business characterization, Business plans
and description. Inputs from SME
e.g. number of users, items in warehouse

Functional characterization
Application components, software system
e.g. EIS systems, database requests,

Resource characterization
Processors, memory, time

Modeling Performance

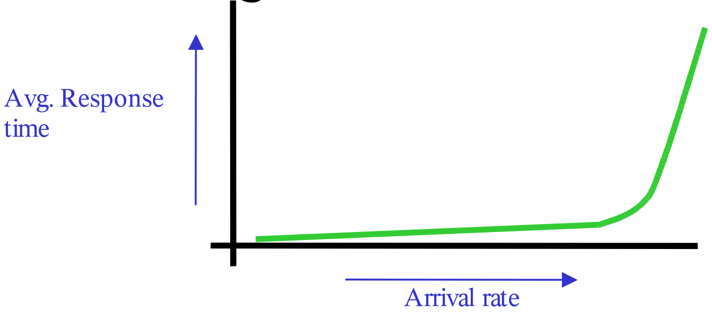
- Representation of how systems & resources are used by workload
- Models at System level
  - Back box only considering throughput
- Models at Component level
  - Consider interactions
- Capture main factors in determining performance
  - Workload parameters
  - System parameters
    - App server cluster configuration, load balancing, number of connections etc
  - Resource parameters
    - Bandwidth, heap sizes, CPU speed & numbers, etc
- Simulation model
  - Mimics behavior of actual system by simulating state transitions
- Analytical models : There's a method to the madness !
  - Specify interaction via mathematical formulas
  - Most developers will ***never*** really use these !
  - Complex math, require a dedicated performance engineer
  - E.g. Zipf's distribution:

    $$P_i \propto \frac{1}{r}$$

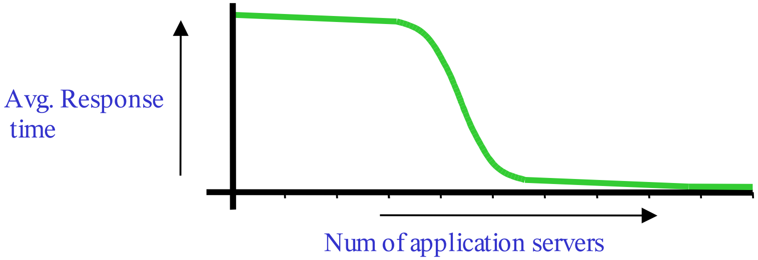
    - Frequency of use of the nth-most-frequently-used word in any natural language is inversely proportional to n.
    - Hot documents, operations, used by search engines for ranking and caching
  - E.g. Markovian chains and discrete time stochastic processes

Example of Modeling

- Web service modeling example
  - A finite QN, infinite population, variable arrival rate
  - Risks: Bad handling of business, under provisioning



  - A finite QN, infinite population, fixed arrival rate
  - Risks : Under or over provisioning



Why Is Modeling Important

- Allow Web service providers to
  - Predict what SLA they can support
  - What SLA they require
  - Evaluate resource allocation alternatives
    - Load distribution, intermediary placement, caching policies
    - Evaluate networking impact
    - Answering what-if questions
      - Change in components, configuration, traffic
      - E.g., Should the app server nodes be doubled
      - E.g. Should the CPUs be replaced with faster
    - Help predict performance

Define Requirements & Gather Metrics

- Define QoS for *your* Web Service
  - Think SMART
    Specific , Measurable, in Agreement, with Responsibility, Testable
- Latency
  - Time between client initiating request and server processing
  - Includes SOAP message marshalling, un-marshalling
- Execution time
  - Time taken by endpoint to perform business task
- Response time
  - Latency + Execution time
  - Viewed from a network node's perspective
- Transaction time
  - Time taken to execute business task,
  - May involve multiple SOAP message exchanges
- Throughput
  - Amount of data processed by the endpoint

Example of Metrics

- Example
  - Web service should processes 9000 HTTP Requests in 30 min with a SOAP response message size of 467 Kb

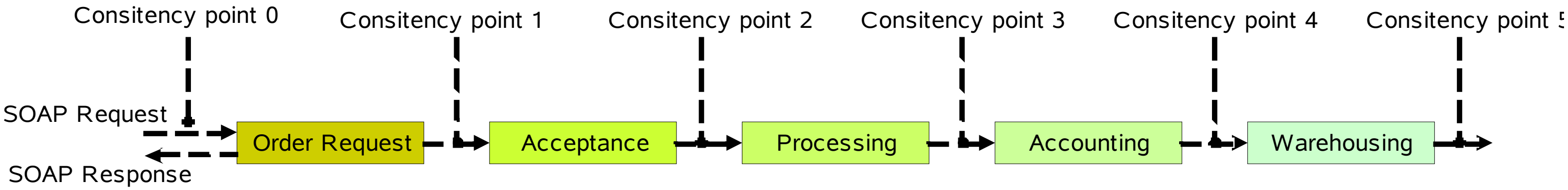
$$Throughput = \frac{Total\ requests * Average\ size}{Time\ period}$$

$$Throughput = \frac{9000*467000}{1800} = 1{,}425{,}39\ Kbps$$

  - Web service node should be deployed on at least a T1 line

Example of Metrics

- Example
  - How do you calculate response time in asynchronous processing for an SLA ?



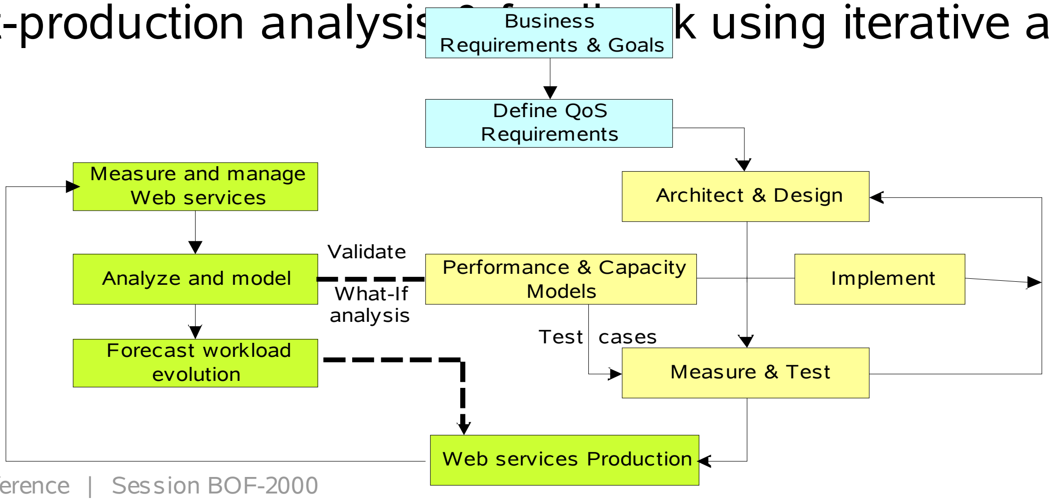
  - Decide on consistency points

$$\mathrm{ResponseTime} = \sum_{1}^{5} \left(\mathrm{Time}_{\mathrm{CP}_i} - \mathrm{Time}_{\mathrm{CP}\ i-1} \right)$$

Source: Gimarc & Spellman CMG 1999

Modeling Performance

- Business and functional requirements
  - Tied to overall vision and evolution plans
  - Describes third party usage, quantitative descriptors
- Architect and design with an iterative approach to mitigate risk
  - Use a quantitative approach to model and test
  - Evolutionary, wire frame prototypes
- Production isn't throwing over the fence
  - Post-production analysis & feedback using iterative approach



Agenda

Scalability & Performance

- Limiting factors for Web services
- Quantitative design and development
- Monitoring and measuring
- Problems & Mitigation Strategies
- Summary

Monitoring and Managing Web Services

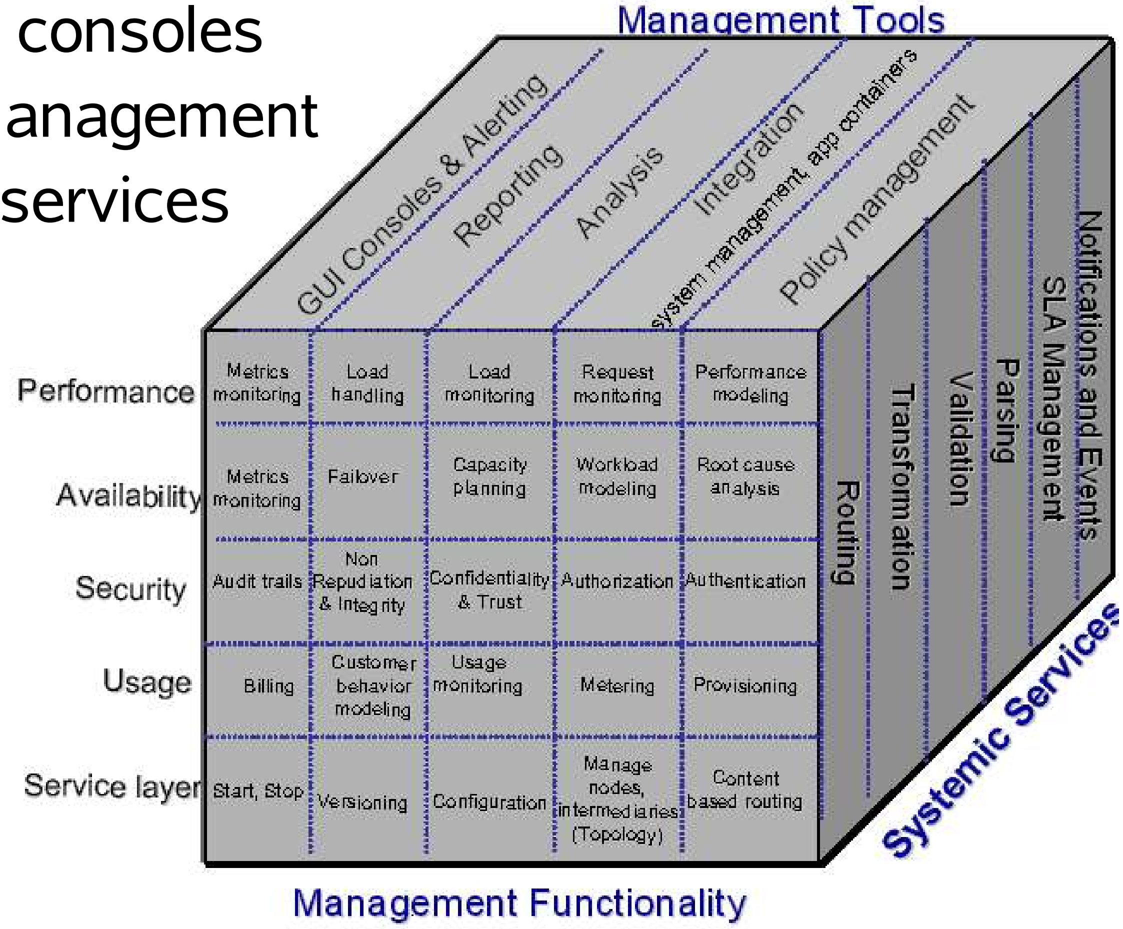
- Monitoring , measuring and management is critical for production
- What may be good for Web services :
  - All the loose coupling
  - Interfaces abstract the details of functionality
  - Application servers obscure the details of threads, transactions, persistence
  - Middleware, ESB's abstract details of distribution
- Good from an engineering standpoint but :
  - Makes performance, capacity modelling, monitoring, management & analysis much harder !

Monitoring and Managing Web Services

- Web services management
  - Common goals as system management
    - Supplement, not replace traditional IT management
  - Works at the Web service layer
  - Intercepts, inspects, and filters messages
  - Transforms, re-routes SOAP/XML message contents to address or prevent problems
  - Analyze Web service performance against SLA
- Online monitoring and control though policies
  - Performance management
  - Fault management
  - Usage management
  - Security management
- Offline analysis, planning, and administration
  - Performance modeling
  - Capacity planning, workload modeling
  - Business analytics, customer behavior
  - Cost modeling (usage metering, chargeback’s & billing)

Logical Components

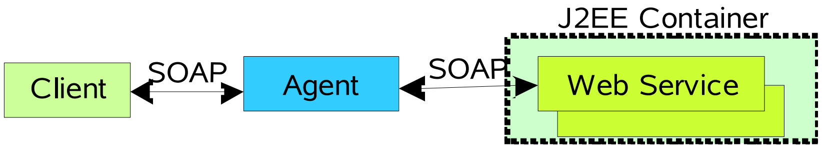
- Logical components
  - Tools and consoles
  - Service management
  - Systemic services



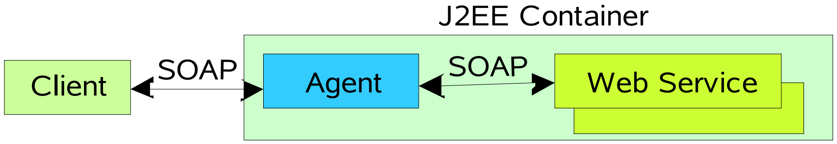
	GUI Consoles & Alerting	Reporting	Analysis	Integration system management, app containers	Policy management
Performance	Metrics monitoring	Load handling	Load monitoring	Request monitoring	Performance modeling
Availability	Metrics monitoring	Failover	Capacity planning	Workload modeling	Root cause analysis
Security	Audit trails	Non Repudiation & Integrity	Confidentiality & Trust	Authorization	Authentication
Usage	Billing	Customer behavior modeling	Usage monitoring	Metering	Provisioning
Service layer	Start, Stop	Versioning	Configuration	Manage nodes, intermediaries (Topology)	Content based routing

Architectural Components

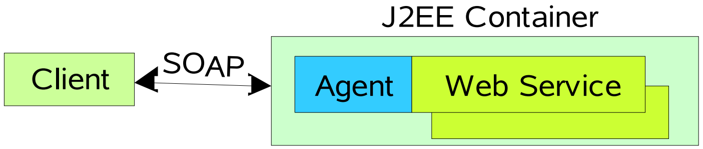
- Proxy based
  - Agent brokers requests, responses
  - Stand alone
    - Simple deployment
    - Can be used as integration point (e.g. SSO)
    - Good for cross domain
    - Additional hop can add latency



- Container based filters
  - Deployed as extension to container & runtime
    - Can exploit capabilities of container (e.g. JMX)
    - Agent specific to container (plug-in)
    - Cross domain problems of control
    - Bad agents can affect the containers performance



- JAX-RPC handlers
  - Deployed as handlers with J2EE Web services
    - Functionally equivalent to filters
    - Perform better



Monitoring and Managing Web Services

- Management standards
  - OASIS WSDM (Web Services Distributed Management)
  - Management Using Web Services (MUWS)
    - Use of Web Services to manage IT resources
  - Management of Web Services (MOWS)
    - Web services as managed resources
- Vendor solutions :
  - Actional, AltaWorks,Amberpoint, BlueTitan, Confluent, Infravio, Santra, Service Integrity, TeaLeaf, WestGlobal and more !
  - Many others
  - Moving to hardware based implementations

Agenda

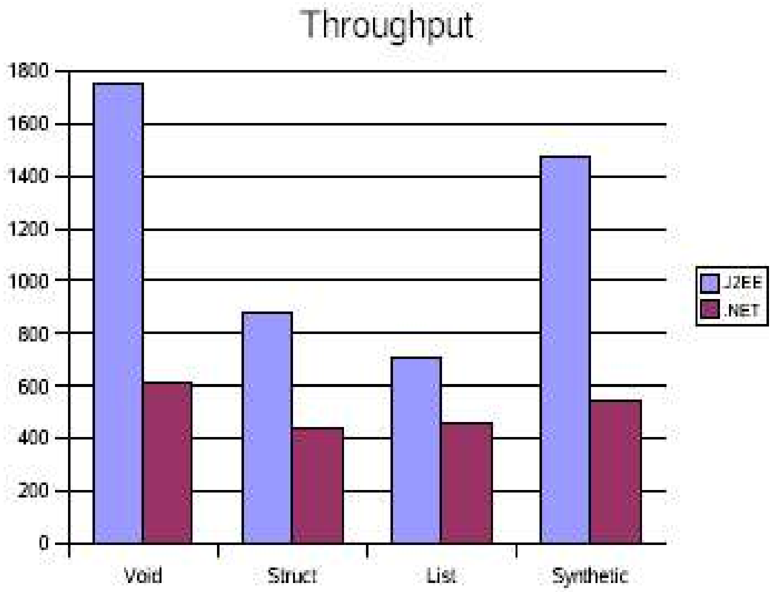
Scalability & Performance

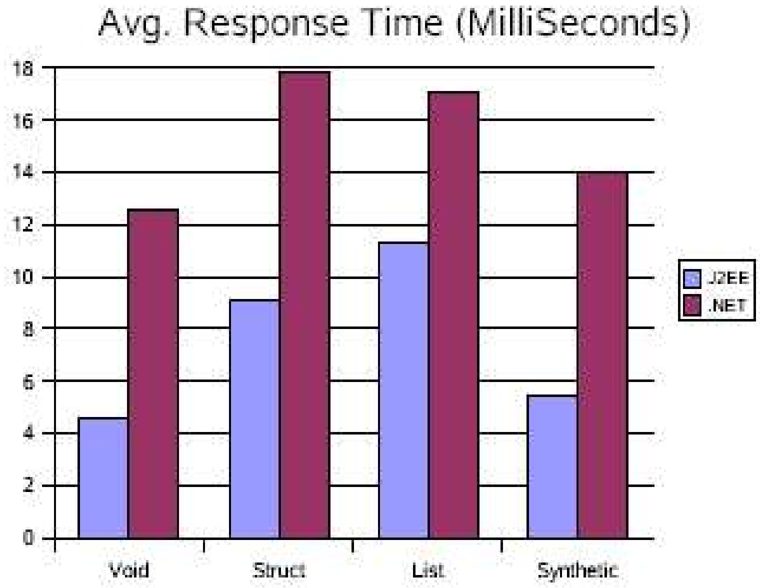
- Limiting factors for Web services
- Quantitative design and development
- Monitoring and measuring
- Problems & Mitigation Strategies
- Summary

Issues And Strategies

- Choosing the right platform
- Choosing the Network infrastructure
- Distribute the processing
- Compression
- Content encoding schemes
- Content processing
- Content based routing
- XML parsing
- XML transformation
- XML validation
- Application design

Choosing the Right Platform





J2EE platform offers better performance and scalability than .NET.

- JAX-RPC performs 3X faster than Microsoft .NET.
- Read more at http://java.sun.com/performance/reference/whitepapers/WS_Test-1_0.pdf

Choosing the Network Infrastructure

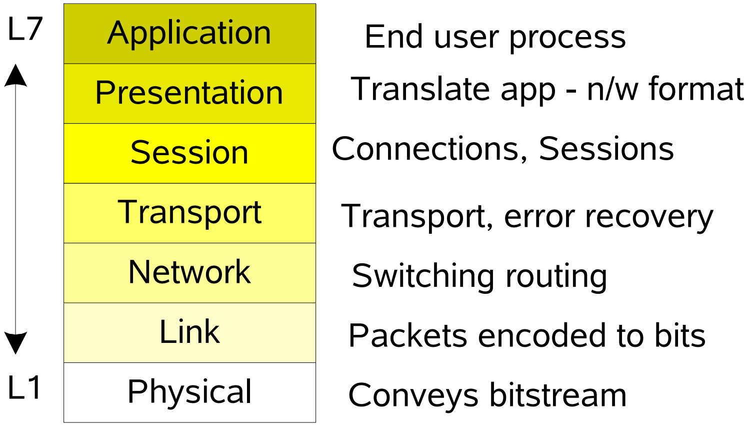
- Web services based transactions demands heavy-weight server infrastruture at the provider.
  - XML traffic is 15-20 times larger in payload than equivalent binary-encoded traffic
  - Vertical scaling infrastructure with support to add more CPUs, Memory, Storage and Gigabit network.
    - Demonstrates and proves better scalability and performance in heavy XML payload and processing.
  - Horizontal scaling requires adds a lot of extra effort in Clustering, Load balancing, Storage and Management.
  - Horizontal scaling through Grid computing architectures shows better scalability
    - For transactions involving multiple intermediaries and processing hops

Distribute the Processing

- Some tasks are CPU intensive
  - XML Digital signatures
  - XML Encryption
  - SSL protocol
- Strategies
  - Adopting XKMS reduces payload offloading key distribution and registration.
    - Delegate public-key lookup/registration/verification tasks
  - Distribute & offload tasks across separate hardware
  - Different from clusters, where servers have equivalent configuration
  - Use hardware based accelerators (For example:)
    - Sun Crypto (4300 TPS,2048-bit RSA, 3DES bulk encryption @ 500 Mbps)
    - nCipher’s SSL (1600TPS 1024-bit, Secure keys,certs in hardware)

Distribute the Processing

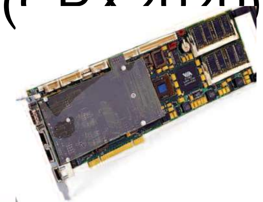
- XML Appliances & Accelerators
- Move cumbersome work to hardware
- Technology is evolving
  - But solutions available today
- Solutions in Layer 7,6,5 & 3



L7	Application	End user process
	Presentation	Translate app - n/w format
	Session	Connections, Sessions
	Transport	Transport, error recovery
	Network	Switching routing
	Link	Packets encoded to bits
L1	Physical	Conveys bitstream

Distribute the Processing

- Layer 6 & 7
- Establish & Enforce policies
  - Policies per-service, partner or transaction basis
  - Integrate into existing security & identity management
- Adjunct or Intermediary
  - Next to application server or as network proxy
- Offload processing
  - TCP, SSL, XML parsing, Schema validation, XPath filtering, and XSLT
  - E.g. JAXP to send requests
- Offload security
  - XML Encryption, XML Signature, WS-Security, SAML
- Routing based on SOAP, HTTP headers, payload
- Data transformation with XSL at wire speeds
- Stand alone, rack mounted, appliance or blade
- Vendors: DataPower (XA35 , XA40), Sarvega (Guardian, Speedaway), Reactivity (2300 Series) Forum Systems (Xwall), Tarari Content Processors (CPX2020)



Distribute the Processing

- Layer 6, Layer 5
- I/O acceleration, compression
- Typically require paired, symmetrical deployment
- Acceleration, compression, and caching
  - Various application-layer protocols sessions
  - Deployment as intermediate-node
  - Rack mounts
- Vendors: BoostEdge (BE200A), ITWorx (NetCelera), Peribit (SR55),Redline(EX 3250)

Distribute the Processing

- XML aware switching products
  - Layer 3
  - Switch and maintain stateful sessions
  - Intercept, inspect, transform, route, switch, and block requests
  - Network load balancing and fail over
  - Inbuilt SSL acceleration
- Vendors:
  Sun iForce VPN/Firewall appliance
    - Check Point VPN-1/Firewall-1 NG software on Linux
    - DES,IKE,RSA, X.509,shared secret, etc

- F5 BigIP switches (5100,5000),Cisco (Catalyst) others

Compression

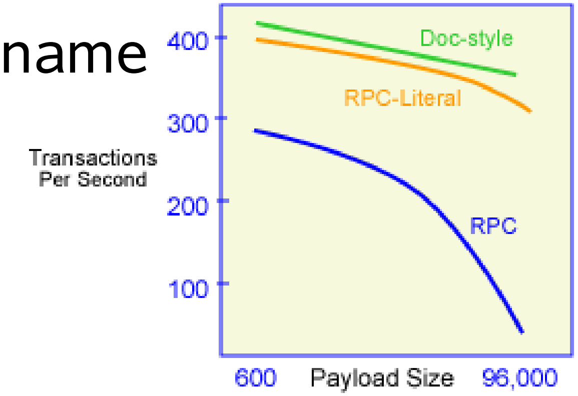
- XML introduces additional layer of abstraction
  - Text based, schema driven
- Compression can reduce latency by packing content
  - Increases CPU load during compression-decompression algorithm processing
  - Increases throughput
  - Typically decreased network latency outweighs processing latency
    - If bandwidth is not a concern wont help much

Compression (cont)

- Strategies
  - GZIP
    - Loss-less, open source, patent free,
    - Standard for Web servers and clients
    - Algorithm uses distribution of common sub strings
    - Large XML docs can exceed 90% compression ratios
    - Static vs. Dynamic compression
    - On the fly compression
      - Vendors : JXEL, Vigos etc
  - Any compression-decompression scheme requires symmetrical deployment
    - Ensure client's SOAP engine support
    - Ensure support for SSL

Content Encoding Schemes

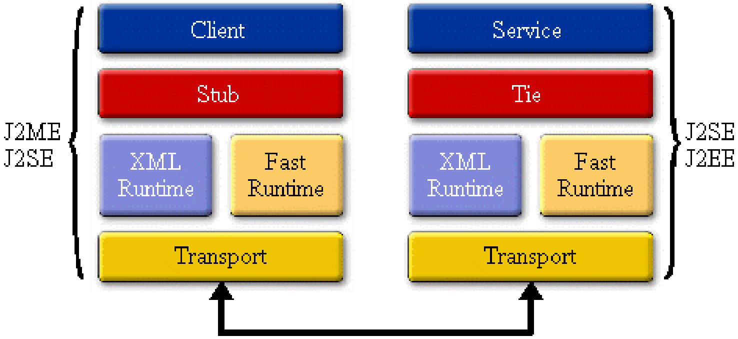
- RPC-Encoded, RPC-Literal
- Document-Literal, Document-Encoded
- Strategies
  - Use Document-Literal
  - Encoded is slower than Literal
    - SOAP Encoding can serialize arbitrary graphs
    - Literal mode limited to trees
    - Data type attributes not inserted in elements
    - Body not wrapped with a method name
    - RPC-Literal ≈ Doc-Literal
    - Literal for interoperability



Source: PushtoTest.com

Content Encoding Schemes (cont)

- Strategies
  - Use alternate encoding schemes
    - ASN.1 (FAST Web Services)
      - ASN.1 Schema for SOAP 1.2
      - ASN.1 Schema for the XML info set
      - Fast annotations for WSDL
  - Attachments
    - SOAP with Attachments (SwA)
      - Uses MIME
      - WS-I Attachments Profile 1.0
    - SOAP MTOM (Message Transmission Optimization Mechanism)
      - W3C effort
      - Model the message using XML infoset
    - WS-Attachments,
      - Uses Direct Internet Message Encapsulation (DIME)
    - JAX-RPC supports attachments today



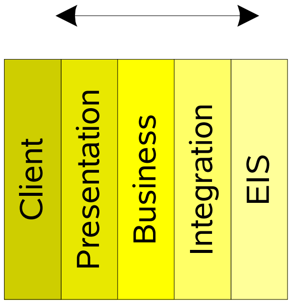
Content Processing

- Consider the characteristics of the services
  - Some use cases require service to generate identical content independent of clients identity
  - Read only types of services
    - Perform queries, retrieve data
  - Some services require other services, data
    - Repeated operations and calls
    - E.g. validation rules, reference data lookups
  - Transformations, style sheets needed repeatedly
- Typically tied to backend EIS systems
  - Limitation on throughput, concurrency, connections

Content Processing (cont)

- Strategies
  - Use caching where it makes sense.
    - What and where to cache
    - Expiration and cleanup policy
    - Data loading policy (MRU, LRU etc)
    - If you need a distributed coherent caches
    - Caching vs pooling
  - Reverse proxy cache
    - Close to servers, content generated by server
    - Use handlers, intermediaries
  - Forward proxy cache
    - Closer to clients
    - Aggregates content from multiple servers
  - Caching different parts of reference data
    - Minimize lookup time, squeeze capacity to max
  - Caching at different tiers
    - Network, web server
    - J2EE caching
      - Application server capabilities
      - **Vendors**: Livestore, Gigaspaces,Tangosol etc

- Caveats : More memory, cache misses, stale data, volatile data, per user data, synchronization across nodes

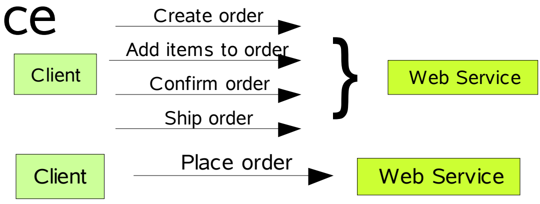


Content Based Routing

- Static routing
  - Use HTTP headers, SOAPAction
- Dynamic routing
  - Enables inspection of message content and re-routing based on rules and policies
- Strategies
  - Integration servers
    - Workflow
    - Vendors: Sun, BEA, Tibco,SeeBeyond, Vitria etc
  - Application data routers
    - XML centric
    - Inline proxy agents
    - Congestion management
    - Vendors : Actional, Amberpoint, Confluent, Infravio, WestGlobal etc

Application Design

- N-Tier architectures (five or more)
- Dependence on EIS systems with latency
- Strategies
  - Client side application logic
    - Web service clients are usually head-less applications
  - Interface design
    - Coarse grained vs. fine grained message exchange patterns
  - Validate inputs and avoid state maintenance
    - Stateful vs. stateless design
      - Session state is not always a good idea
      - Use correlation ids



  - Proactive
    - Design with performance in mind (Java,J2EE,XML etc)
  - Definitive : Requirement driven
  - Reactive: Test – Analyze – Correct
  - Vendors : Parasoft, Emperix, RadView,Segue, etc

XML Parsing

- SAX, DOM, JDOM, DOM4J, Crimson, Xerces, Electric XML, Pull Parser and more
  - Benchmarks available E.g. sosnoski.com
- Strategies
  - Use SAX when
    - Need to serially access XML elements
    - Need to process parts of documents
    - To process the document only once
  - Use JDOM when
    - Document model fits the core data structure of application

XML Parsing (Cont)

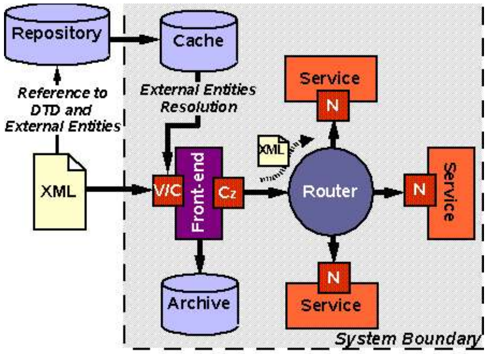
- Strategies
  - Use lazy DOM mode (if parser supports it) when
    - Processing only parts of DOM tree
    - For example, Xerces supports this via *defer node expansion* feature
    - Pros : Construction of DOM tree completes fast
    - Caveat : Parser specific functionality
  - Interrupt parsing
    - All the needed information has been extracted by throwing an **`EndOfProcessingException`**
    - Pros : Efficient when the specific information is needed from a large XML document
    - Caveat : Does not help when the information is located at the end of the XML document

XML Transformation

- XSLT usually takes CPU and memory
- Can be expensive at runtime
- Strategies
  - Use XSLTc
    - Compiles style sheets into byte code
    - No code modification, can be used under JAXP
  - Cache style sheets: **`javax.xml.transform.URIResolver`**
  - Use hardware to handle transformation

XML Validation

- Complex XML schema & data structure validation takes CPU, time



- Strategies
  - Turn *on* the validation when
    - Documents cross system boundaries
  - Turn *off* validation when
    - Documents exchanged within the system boundary
      - Consider using objects, JAXB instead
    - An XML document has already been validated once
  - Use canonicalized documents where possible
  - Reduce cost of referencing external entities
    - Use Standalone documents, Proxy + setEntityResolve

Agenda

Scalability & Performance

- Limiting factors for Web services
- Quantitative design and development
- Monitoring and measuring
- Problems & Mitigation Strategies
- Summary

Summary

- Performance and scalability is essential for mature production level services
  - Cross enterprise communication, business risks of ignoring
  - Simplicity helps with interoperability & performance
- Performance is an on going exercise
  - No magic bullet
  - Modelling and analysis go hand in hand with architecture and design
- Strategies, solutions are available today
  - Application level considerations
  - Web services management for monitoring
  - Content aware routing and prioritization
  - Hardware based acceleration

Thank You !

- Source for Java & XML
  - java.sun.com/xml
- Interesting benchmark suites
  - XML processing www.sosnoski.com
  - Encoding and others www.pushtotest.com
  - XSLTMark www.datapower.com
- Speakers
  - Sameer Tyagi (s.t@sun.com)
  - Ramesh Nagappan(ramesh.nagappan@sun.com)