NetWeaver Performance: A Customer Point of View

Joerg Nalik

NetWeaver Solution Office

SAP LABs, LLC., Palo Alto

April 2005

THE BEST-RUN BUSINESSES RUN SAP

Overview

Through this presentation you will learn:

- **How customers view performance.**
- **How the customer's view on performance links up with development criteria for performance.**
- **How to define performance.**
- **Tips for design of well performing NetWeaver applications.**
- **What NetWeaver, SOA, ESA means for application performance.**

THE BEST-RUN BUSINESSES RUN SAP

Different Definitions on Performance?

	SAP Developer	Customer
Guidance	PIL process	NetWeaver, SAP SOA, ESA Adoption Programs
Documentation	SAP Performance Standard	"How to ...." papers about performance
Criteria	Persistence Layer ➢ Indexes ➢ Where clause ➢ Buffer/Cache ➢ Not multiple accesses with identical results Application Layer: ➢ enable parallel processing ➢ Linear dependencies ➢ 2 application roundtrips per user interaction Other: ➢ average response time below 2 seconds ➢ Sizing procedure available	3S2R: ➢ Sizing Landscapes ➢ Scalability of system landscape ➢ Stability ➢ Response Times ➢ Robustness

Different Definitions on Performance? (cont.)

Development:

➢ The SAP Performance Standard for SAP Developers gives concrete advise on how to design applications and coding for good performance (DO and DON'T list).

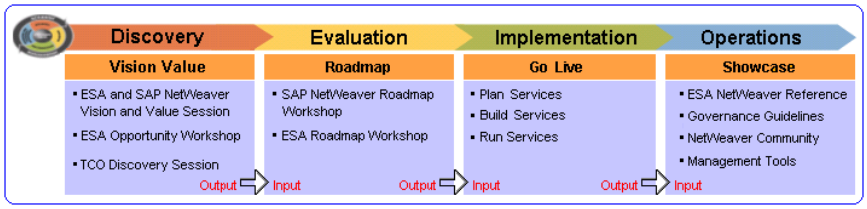
➢ Bottom up approach:
- ➢ Re-usable modules, libraries, APIs ....
- ➢ platform components
- ➢ application components
- ➢ composite applications
- ➢ business scenarios
- ➢ Solution packages

Customers:

➢ need to define performance in more concretely measurable and verifiable terms in order to match them up with business requirements.

➢ Top down approach:
- ➢ Business need, Budget, Resources, TCO/ROI, SAP or none SAP....
- ➢ IT landscape, extending and integrating existing solutions
- ➢ business scenarios
- ➢ hardware, network and software components
- ➢ custom development

Performance within the ESA Adoption Program



- **Discovery/Evaluation:**
  - **Collect performance business requirements (volume, concurrent users, response times, availability)**
  - **Translate business into technical requirements**
- **Implementation:**
  - **Design IT landscape for performance**
  - **Tuning according to "How to ...." papers**
  - **Load Testing for 3S2R verification (and trouble shooting if necessary)**
- **Operation:**
  - **Production Monitoring for verification of performance "Service Level Objectives".**
  - **Production Monitoring for pro-active continuous landscape capacity management.**

Ref: https://www.sdn.sap.com/sdn/services.sdn?node=branchn3-1&contenttype=url&content=%2Fhtml%2Fesa_welcome.htm

Sizing

How to:

- **Get specific customer's requirements:**
  - **either transaction volume or concurrent users + Think Time**
  - **maximum allowed response time for international WAN users**
  - **exact, best screen by screen, scenario definitions**
  - **....**
- **Get SAP reference benchmark data for the scenarios the customer wants.**
- **→ calculate landscape needs: hardware, network, failover, security components.**

The Real world situation:

- customers cannot specify requirements with sufficient accuracy
  - brand new scenario, no experiences, no references
  - no production monitoring data from legacy systems
  - Internet facing: number of users and usage patterns unknown
- customers cannot define specifics of scenarios and/or scenarios are highly variable (browsing, size of result sets,…)
- SAP benchmark data for the scenarios an individual customer wants are not available. Every customer runs different customized scenarios.

An Alternative Approach: Continuous Capacity Management:

- very rough first sizing estimate based on requirements and benchmark data or reference customer data.
- refining sizing based on concrete customer specific load testing results (after a landscape tuning exercise).
- pro-actively adapting landscape sizing during production based on production monitoring data.

Sizing: What does it take to design a high end system

Scalability:

- **linear dependency: costs for doing something should depend linear on result set size, concurrency volume and similar. Otherwise TCO < ROI is not achievable in high end systems.**
- **enable parallel processing:**
  - **Ability to cluster component instances or work with multiple instances in parallel.**
  - **multi-connect and load balancing to different instances in a chained component landscape.**
  - **connection pooling between application components in particular for synchronous communications.**
  - **no or only little locking/synchronization between parallel application processing.**

Sizing: What does it take to design a high end system

TCO < ROI:

- **efficient access to the "back-end" component which serve as "persistent layers".**

Watch for upper limits:

- **data set sizes cannot grow indefinitely: A large result size might not fit into process memory heaps, in particular in Java-components. Example: 1000nds of line items in a purchase order send in between components as XML/soap message.**
- **partition large processing chunks.**
- **Delivery of results to the browser is subject to a 120 second timeout. Use progress bars or other busy signs to extend timeouts and keep the end-user informed.**

Sizing: Examples

Customer 1:

- **300,000 named users (grocery stores in the US)**
  - **assume 50 clicks per week per user during a 5x10 working week → 83 clicks/sec average**
  - **assume 60 sec. Think Time → 5000 concurrent users average**

Customer 2:

- **2000 concurrent users in a call center**
  - **10 sec Think Time + some server time → about 150 clicks/sec average**

Customer 1 has a very high concurrently open session load. Each session needs memory even if no request is processed. Therefore, pay attention to session memory sizing.

Customer 2 has a high transactional load. CPU sizing is most important.

Sizing: Examples

Customer 3:

- **Web-service calls delivering back 1000-2000 result lines via a XML/soap message which is then processed in a WAS 6.40/Java component:**
  - **number of result rows: 1600**
  - **XML file size: 16 MB**
  - **GC memory turn over: 1200 MB**
- **→ 1 server node provides usable memory heap for only 10 concurrent users.**
- **→ response times might exceed 120 sec.**

Possible solutions:

- **re-evaluate business need for sending >1000 line items.**
- **massive Java server node clustering to allow larger usable heap memory**
- **Replace DOM XML parser by SAX parser for more efficient use of memory.**

Note: XML is convenient but large XML-messages need a strong business case to pay for the hardware expenses needed to process them!

Stability

Criteria for Stability:

- How long can an application server run without needing to be restarted or without a server crash?
- What percentage of user sessions/transactions are allowed to be lost or resulting in error messages to the end user?

Typical stability problems:

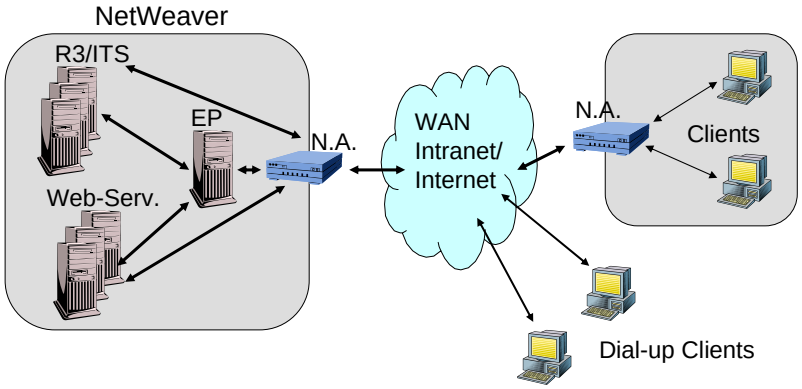
- memory leaks in Java
- memory leaks in exception and error handling and logging
- connection losses between NetWeaver components
- insufficient tuning, running out of pooled resources (threads, db connections, other remote connections ….)
- no logoff, session closure issues, issues propagating session closure to backend systems
- Out of Memory (OOM) problems due to large objects processed in memory.
- insufficient hardware sizing or un-balanced hardware sizing of some NW components
- incorrect or no load balancing in a cluster/multi instance set-up, wrong request routing stickiness to server nodes

Stability

Design for stability:

- limiting and managing of memory data caches in Java
- re-try and re-connect mechanisms
- partitioning of processing of large objects
- choosing most efficient parsing algorithms in regard to CPU and memory consumption
- clear user session concept across NW components
- never forget to provide a Logoff button and a mechanism to close sessions.
- proper freeing of resources (memory, connections, threads) also in exceptional or error situations.

Customer's NW landscape



A typical NW landscape consists out of one EP instance plus usually multiple web enabled applications such as SAP ERP systems with ITS or others. EP facilitates single sign-on and routing to all web-enabled applications. However, web enabled applications might be linked to by re-directs (URL-Iviews) and send content directly back to clients, bypassing EP. In front of EP there are typically Network Appliances (NA) like load balancers, proxies and firewalls.

Response Time

Common NW use case of customers:

- **exposing ERP backend data to new user groups over the internet**
- **Collaborative business scenarios**
- **SOA/ESA approaches**

Challenges:

- **long Wide Area Network distances**
- **https required**
- **Many synchronous roundtrips:**
  - **Browser – Portal**
  - **Browser to Portal/middle tier components like WAS 6.40, ITS**
  - **middle tier components to backend systems**
- **Web Service force use of XML based data transfer (vs. more efficient proprietary protocols)**

WAN multipliers

LAN:

Server time, Browser time 2+1 sec. = 3 sec

Intranet:

LAN + <http roundtrips> * <network latency time> = 3 + 5*0.2sec = 4 sec

Internet:

Intranet + Netw. Appliances Time + <data volume/TCP-IP package size> * <network latency time> = 4 sec + 1 sec + 20 * 0.2 sec = 9 sec.

Internet + "weak clients" (http 1.0 only):

Internet + (<https> + <connect> roundtrip) * (<netw. Latency t.> + <server user authentication time>) + <no compression-more data roundtrips> * <netw. latency time>= 15 sec + 10*0.2 + 5*0.3 + 20*0.2 = 22.5 sec

Internet + "weak clients" + low network quality-package losses:

Internet + weak clients + <resend factor> * <all roundtrips> * <network latency> > 30 sec up to minutes

Customer:

"It works with our other (none SAP) applications"; "Yahoo, Amazon, Ebay, …performance is fine" ….

Response Time

Development Design considerations:

- Server time vs. Network latency times optimization. For instance delta management approaches to information transfer increase server times for the benefit of reducing network times. DO WAN TESTING!
- minimize the number of roundtrips
- use caching, buffering along the chain Database, ERP, middle tier, Portal, Proxies, Browsers
- minimize data volumes
- consider "syndicated, federated" architectures and concepts:
  - local regional deployments of the application for reducing network latency times.
  - regular data replication between globally distributed instances.

Robustness

Definition:

If a NetWeaver landscape is temporarily overloaded for some minutes performance might degrade but servers should not crash and the landscape should fully recover when overload ends with no need for re-starts.

Design considerations:

→ See Stability

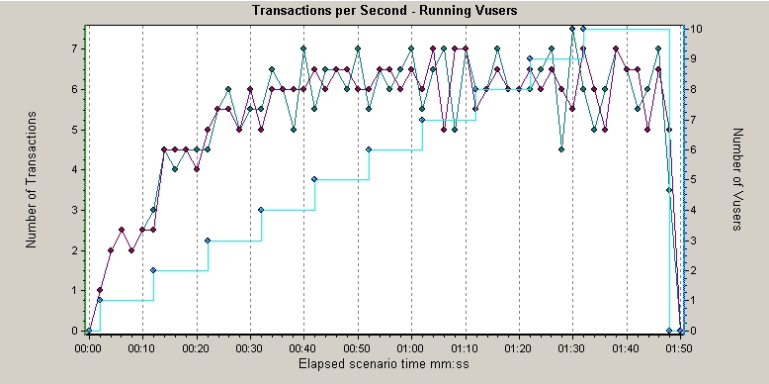
How to test

- **Use a load testing tool. SAP recommended is Mercury LoadRunner.**
- **If possible, do some testing over a WAN and with https.**
- **Load test methodology, see:**

How to Perform SAP Enterprise Portal Load Testing on SDN

Example: Ramp-up testing

- **Ramping user load up over time measures:**
- **Capacity: here 2 * 6 req/sec = 12 req/sec**
- **Scalability: Verify that at 12 req./sec CPU usage is close to 100%**
- **Robustness: Even if you add users when already at maximum capacity no component in the landscape should crash.**
- **Example done with Mercury LoadRunner, 10 virtual user, 0 Think Time**



The near future

- **Expect more and more applications needed to be Internet enabled.**
- **Rise of SOA, ESA applications using web services/ XML messaging between peer application components.**
- **Component testing is less then half the job. Application integration tests become more and more important.**
- **Rise of IT Governance: Cross departmental project, budget, and resources planning.**
- **"Ebay" like requirements:**
  - **high numbers of concurrent users**
  - **performance optimization for WAN**
  - **lean client side rendering**
  - **still richness of content and functions.**

If a web application does not perform end-users go elsewhere and business is lost.

Q & A