Enterprise Application Performance Factors

Do, Don’t and Best Practice

Throughwave Infrastructure Engineer Certification Course Level I

Agenda

- Overall Enterprise Application Performance Factors
- Best Practice for generic Enterprise Application
- Best Practice for 3-tiers Enterprise Application
- Hardware Load Balancer
- Basic Unix Tuning
- Performance Tuning Tools

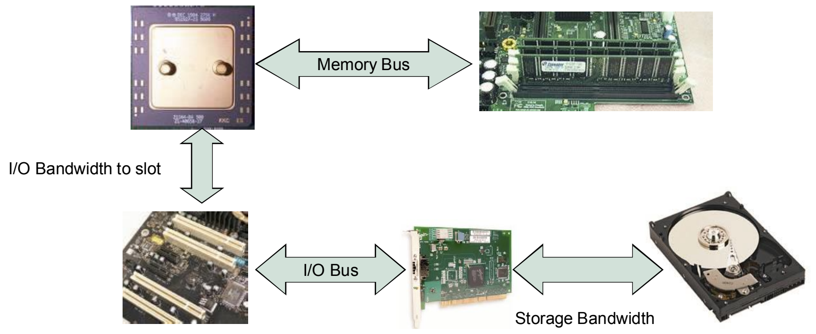
Current form of Enterprise Applications

- Console Based Application
- 2 Tiers Application (Client-Server Model)
- 3 Tiers Application (Web Application, Application Server Model)

Factors to Application Performance

- Running Platform
  - Architecture of Hardware Platform
  - Architecture of Operating System
- Operating System Architecture
- Application Parameters

Hardware Architecture

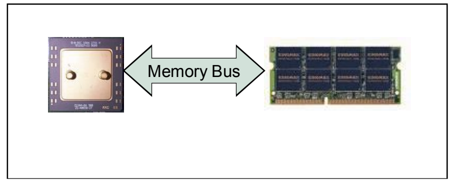


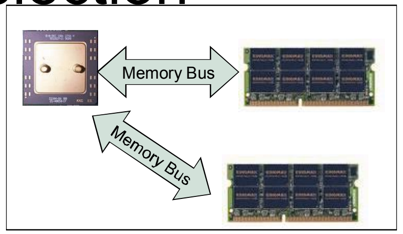
- Four Things to remember
  - CPU Performance is not everything
  - Memory Bandwidth
  - I/O Bandwidth
  - Storage Bandwidth

Software Parameters

- Operating System Architecture
  - Normally Application dictated which platform we will run our application on.
  - Tuned Operating System always faster than default installation.
  - Windows or Unix does not make different much after tuned – However..
- Application Parameter
  - Tuned Application always faster than default installation.

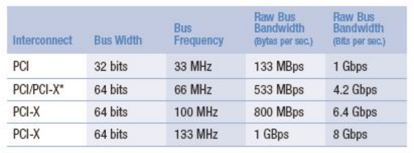
Hardware Selection





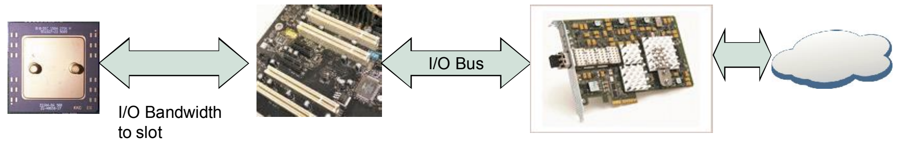
Do:

- Pick the most balance CPU architecture possible
  - Multiple CPUs, Multiple Memory Buses
- Pick server with fastest I/O Bus possible
  - PCI-Express >> PCI-X >> PCI
  - PCI-Express is serial bus, PCI-EX x16 bus is faster than PCI-EX x 2 bus.
- Pick the most suitable memory speed for each CPU
- Use enough memory to satisfy swapping space allocation for target Operating System



Interconnect	Bus Width	Bus Frequency	Raw Bus Bandwidth (Bytes per sec.)	Raw Bus Bandwidth (Bits per sec.)
PCI	32 bits	33 MHz	133 MBps	1 Gbps
PCI/PCI-X*	64 bits	66 MHz	533 MBps	4.2 Gbps
PCI-X	64 bits	100 MHz	800 MBps	6.4 Gbps
PCI-X	64 bits	133 MHz	1 GBps	8 Gbps

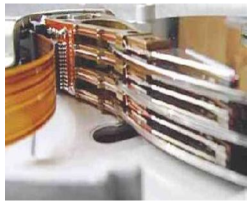
Network Performance Problem

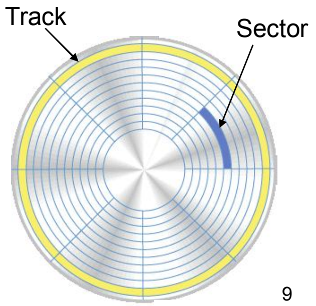


- Select the right network I/O
  - What is the Application bandwidth requirement
    - Is the network controller fast enough?
    - Does it plug into the fastest bus available on the hardware platform?
  - Does application bandwidth need Jumbo Frame?
- DNS problem is the most common performance problem when involving network application
- Routing Setup
- Upstream Switch Capacity

Disk

- Platter
- Track
- Sector
- Cylinder
- Spindle
- How to get to data?





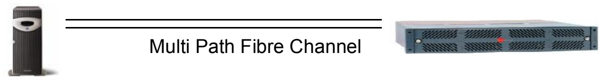
Block Size

- Block size = I/O size unit for that particular disk.
- Every disk transfer combined from control and data transmission.
- Large block I/O means less overhead for storage command.
- The large the block I/O size, the faster the storage bus get saturate
- Sequential access pattern need large block size, but also need to be careful on bus saturation status
- Random access pattern – For every access, disk must seek before I/O operation – so we must minimize seek time instead

Minimize Seek Time

- The larger the disk block, the less the seek time. (less block to request)
- Outside cylinder are faster, disk rearrange/compact usually move data to outside cylinder.

Storage Selection



Do:

- Controller Card from server must be the same bus type as the fastest bus on that server
- If it's internal disk storage, select bus architecture that have the least bandwidth sharing
  - SAS << U320 SCSI << SATA << IDE
- If it's external storage, select external bus that can do multipath and load balancing
  - SAN: Fibre Channel, SAS
- Choose the correct RAID level for application
  - Use RAID 5 for application that read more than write, archiving, data that can get back from backup
  - Graphic for web application RAID-1 is more than suffice.
  - Critical Database Application always using RAID-10
  - Raid-6 is the next storage Raid level – survive two disks failure – striping to the max.
  - The more disk in RAID set, the faster your application would be.

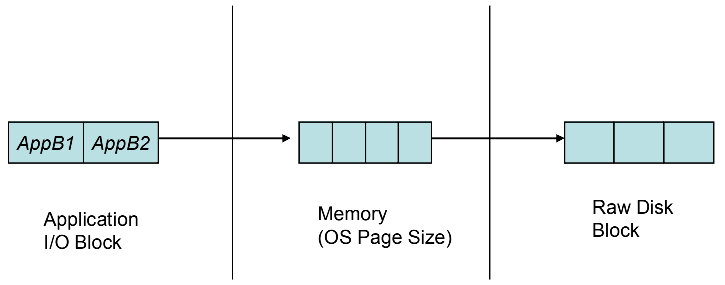
Don't:

- Do not use NAS for critical database application (Exception is Parallel NAS system).
- Do not use RAID 5 for critical application
- Do not use iSCSI for critical application

Solid State

- Fast but expensive
- How fast?
  - 240 MB avg read time
  - 180 MB avg write time
- Compare to
  - 30 MB avg read time
  - 15-20 MB avg write time
- How expensive?
- Where would we using it?
  - Caching tier
  - OS Swap space
  - Database transaction store (Database Log)

Data Block Conversion Problem



- Data move from application to storage system in block fashioned
- Performance could degrade if the system has block conversion overhead
- That's why we need to match block size for all application, OS, and storage block size.
- The rule is force by common dominator from OS page size – common 4KB, 8KB on system like Dec Alpha, Sun UltraSPARC

Basic Tuning Strategy (Best Practices)

Step 1: Server Building Block Foundation Tuning

- Analyze first, if the Application performance could be I/O bound or CPU bound.
- Pick the right component for each server building block
- Select the right storage set up for application type
  - Internal/External
  - Raid Level
  - Storage cache size
- Match Application I/O block size, OS paging size, and real storage block size
- Define storage block size with average largest block I/O size
- Tune virtual memory usage model of Operating System

Step 2: Application Parameter Tuning

- Each application type have different tuning strategy’
- Adhere to recommendation from Application Vendors.
- Basic Rules:
  - Separate Application Tier from Database Tier to minimize impact on Database access model and to allow us to expand with clustering
    - For small size database or embedded application, keeping database tier around always faster than separate it out.
  - For database, try to separate Database Log and Database Data I/O to different physical storage.

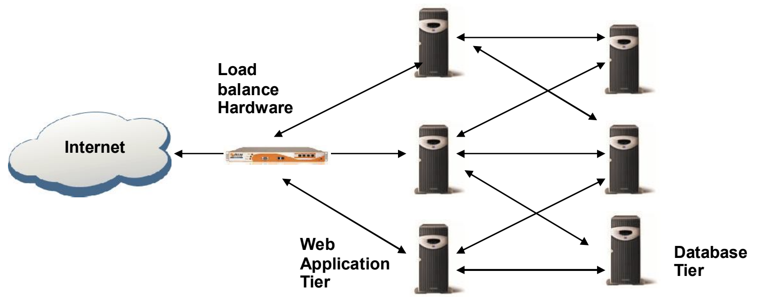
Step 3: Clustering

- In 2-Tier, 3-Tier or N-Tier application model, clustering is the last strategy we use to overcome hardware performance limitation.

How many Tiers can we have?

- Two-Tiers
  - Presentation separate from Business Logic
  - Business Logic and Database Logic still integrated together
  - Ex: Classical Client-Server system
- Three-Tiers
  - Presentation separate from Business Logic, and Backend Tier usually Database Logic
  - Ex: Web Application Server with external database Tier
- N-Tiers
  - Multiple factoring of business logic and presentation logic, most Tier interconnected with middleware logic
  - Ex: Distributed Database Application

How to improve N-Tier Performance



- Clustering Database Tier to help distribute database transaction load
- Clustering Application server Tier

What we need to know before doing any Application Clustering

Database Clustering Facts

- Database look easy to cluster, since all operations are atomic
- However, to do database clustering, we must make sure that all database updates are applied to all database nodes

Application Clustering Facts

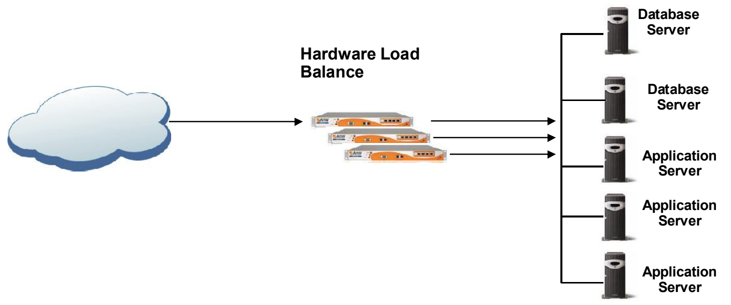
- Request Level Failure – if current node crashes midway, the subsequent requests must be handled on different node automatically
- Session Level Failure – if current node crashes midway, the users that currently login must be able to continue their sessions without losing session states.

Clustering Strategy

Clustering Approaches

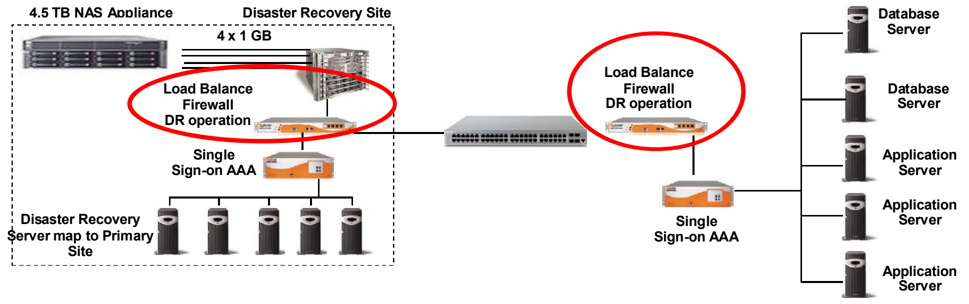
- Database Clustering Approaches
  - Use shared file system and clustering software together to achieve database virtualization, or
  - Use database replication feature to eliminate shared file system
  - Use load balance to distribute load among database cluster nodes.
- Application Clustering Approaches
  - Replicate application logic to all nodes
  - Use load balance to distribute load among application cluster nodes
  - Replicate session information to all application nodes
    - Memory-to-Memory Replication
    - Use centralized shared file system for session replication
    - Use database tier to save session information.

Load Balancing



- Load Balancing helps distribute load among cluster nodes
  - Basic Round Robin
  - Weighted Round Robin
  - Shortest Response Time

Global Server Load Balance



- GSLB can fulfill promises of Disaster Recovery
  - Disaster Recovery needs replication of application and data at the same time.
  - GSLB can help automate request to backup site automatically.
  - GSLB can also do active/active load balance between site.

Load Balancing Alternatives

- Software Load Balance

  Advantages:
  - Inexpensive
  - Simple

  Disadvantages:
  - One point of failure
  - Limited concurrent connections.
- Hardware Load Balance

  Advantages:
  - Automatically handle of session with Web Application
  - Virtual IP eliminate one point of failure
  - Proposed Built hardware can handle much higher load of concurrent connections
  - Ability to do other Application Acceleration features like Compression, SSL acceleration, Link Load Balancing
  - Better ROI in long term

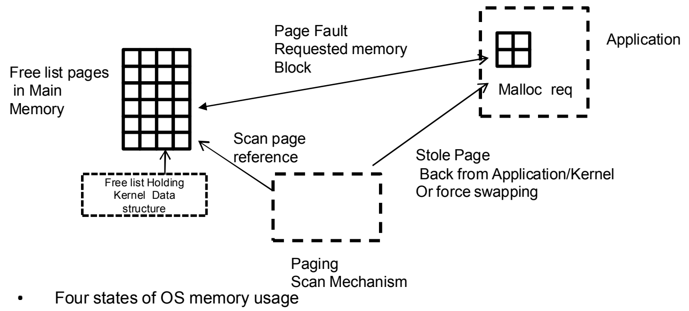
  Disadvantages:
  - Higher startup cost compare to software load balance solution.

Basic Unix Tuning

Disk Pattern

- Partitioning is bad
  - Too many hotspot – move from one partition to other partition on same disk mean issue extra seek command
  - Instead of direct serving data, disk subsystem busy serving seek requests.
- If you can not avoid partitioning (one disk system)
  - Partition/Slice creation have order from outside cylinder to inside. (Mean first partition is the fastest)
  - /root, /swap, /var, /home, /usr sound like something easily optimum.

Memory Tuning



- Four states of OS memory usage
  - Sufficient memory is available, optimum performance
  - Memory is constrained – Paging scan start attempt to stole page back from process
  - System is short of memory – system start swapping -- interactive process response time suffer
  - Memory is running out – Swapping/Paging activity reach saturation
- Increase limit of paging I/O per second to match with I/O bandwidth to paging device (SWAP space)
- Put swap on dedicated fastest I/O if possible – multiple striping disks with hardware controller sound like a good idea.
- Swap space sizing? The most optimum need to measure on target machine under stress.
  - Trigger tuning at half of current swap space.
- Put swap on the fastest slice as possible (except for /root partition)
- Configure multiple raw swap space per system (on different physical disk to create stripe effect for swap space

Questions that need answer?

- What is the I/O access pattern?
- Is the hardware suitable to applications?
- I/O bottleneck?
- Memory access pattern is optimum?
- Processor overload?
- Is processing time in kernel space more than user space?
- Is the contention come from multi-thread blocking/locking?

If the cpu load never increase?

- Load too little?
  - Generate more load and see the relationship of CPU usage
- Application is I/O bound and bottle-neck
  - Check iostat to see what happenning there.
- Externally bound to other services ex: network file system, database
  - Profile the code to see where time was spent?
- Lock contention
  - Profile the code to see where time was spent?
  - Do we use exclusive lock in our code? If so try to see if we can use read/write lock or atomic variable instead

Basic Application Tuning

- Identify data flow pattern
- Keep minimum memory footprint, avoid creating too many objects in the system.
- Object pooling is difficult to make it right, and consume extra overhead for housekeeping and locking shared object in pool
  - retest our code to compare no object pooling with object pooling enable. With hotspot technology in Java 1.5 up, create new object in Java is about as fast as “malloc” in C.
- Minimize logging
  - Console operation is always expensive, minimize logging for our own good. Or check all logging code to allow enable/disable facility at will.