Linux Performance Tuning

Training Material

Atlantic LINUX

License

Except where otherwise noted, this work is licensed under
http://creativecommons.org/licenses/by-nc-sa/3.0/

Atlantic LINUX

Contents

Atlantic LINUX

Contents

- Performance Tuning
  - Web Servers
  - File & Print Server
  - Database Servers
  - Java App Server
- Tuning C
- Tuning Java
- Application Profiling
- Performance Tuning Exercise 4
- In closing ...

Atlantic LINUX

Performance Requirements

- Establish performance targets
- Map targets to business requirements
- Performance comes at a cost
  - More performance comes at a much higher cost
  - Identify *low hanging fruit*
  - How good is good enough?
- Understand your system and its bottlenecks
  - Only ever tune bottlenecks!
- Measure system performance
  - Repeatable benchmarks
  - Tuning without measurement is pointless
  - Record before and after

Atlantic LINUX

Measuring Performance

- Baseline your system
  - Identify important metrics
  - Measurements should reflect performance requirements
- Monitor performance after tuning
- Monitor performance over time
  - Identify performance problems which only occur over time
  - Flag environment changes which cause problems
- Measurement tools
  - Depends on the metrics you want to measure
  - Standard Linux commands
  - Roll your own
  - Off the shelf benchmarks

Benchmarks

- Key characteristics
  - Repeatable
  - Consistent
- Allows you to identify,
  - System statistics that are constant from run to run
  - System statistics that change slightly from run to run
  - System statistics that change dramatically from run to run
- Component or Microbenchmarks
  - Measure standard OS characteristics
  - Useful for comparing systems and tuning OS components
- Application or Enterprise benchmarks
  - Specific benchmark for your system
  - Expensive to develop and maintain

Microbenchmarks

- OS
  - Lmbench
  - Re-AIM 7
  - SPEC SDET
- Disk
  - Bonnie/Bonnie++
  - IOzone
  - Iometer
- Network Benchmarks
  - Netperf
  - iperf
  - SPEC SFS
- Java Application Benchmarks
  - Volanomark
  - SPECjbb
  - SPECjvm
- Database Benchmarks
  - TPC
  - OSDL tests
- Webserver Benchmarks
  - SPECweb
  - TPC-w
  - SPECjAppServer
  - ECPerf

Performance Tuning Exercise 1

1) Download the **lmbench** benchmark and run it on your system.
2) Pair up with someone else and download and run the **Iperf** benchmark between your systems.

Atlantic LINUX

Tuning Guidelines

- Dos:
  - Change one parameter at a time
  - Run your benchmarks after each change
  - Keep records of benchmark runs for comparison
  - Tune bottlenecks
  - Make the least expensive changes first
  - Try to understand your system and how it should respond to changes
- Don'ts:
  - Don't tune without benchmarks
  - Don't tune on systems that are in use
  - Don't assume a change makes things better without measurements

Atlantic LINUX

Hardware Performance - General

- Scale up - buy more hardware
  - add more memory
  - add more/faster processors
  - add more storage
- Consider scaling out
  - increases system capacity rather than speed
  - only suitable to some applications
- Do you need a faster car or a bigger truck?
  - speed or capacity?
- Performance trade-offs
  - removing a bottleneck may cause another one

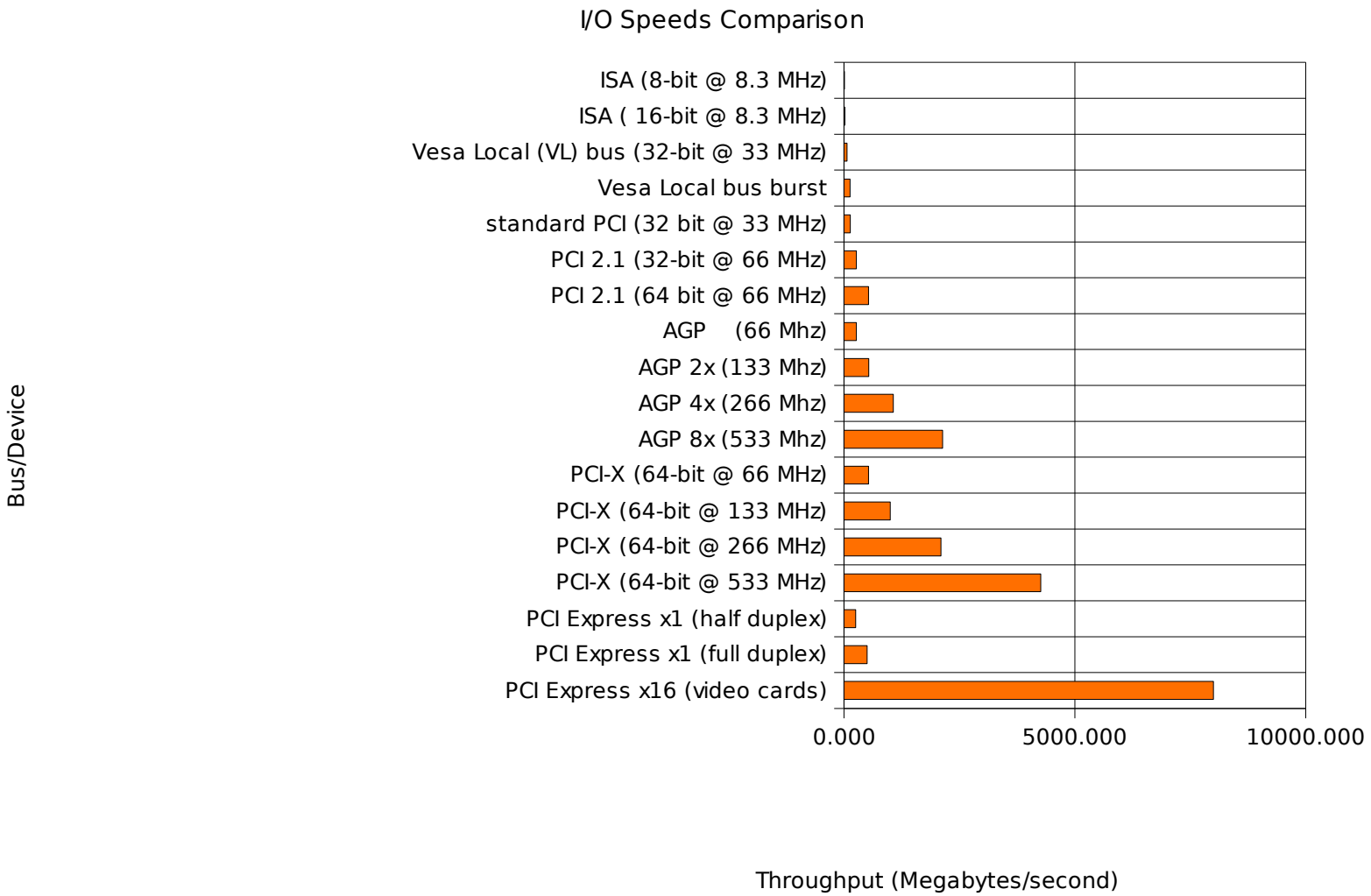
Hardware Performance – Processor

- Speed
  - Processors double in speed every 18-24 months
  - Faster processors are better (if you use them)
- SMT
- SMP
  - multiple processors or processor cores
  - ensure your kernel is SMP-enabled
- Powersaving
- 64-bit
  - address more memory
  - more registers
  - enhanced media processing
  - ensure you are using 64-bit kernel

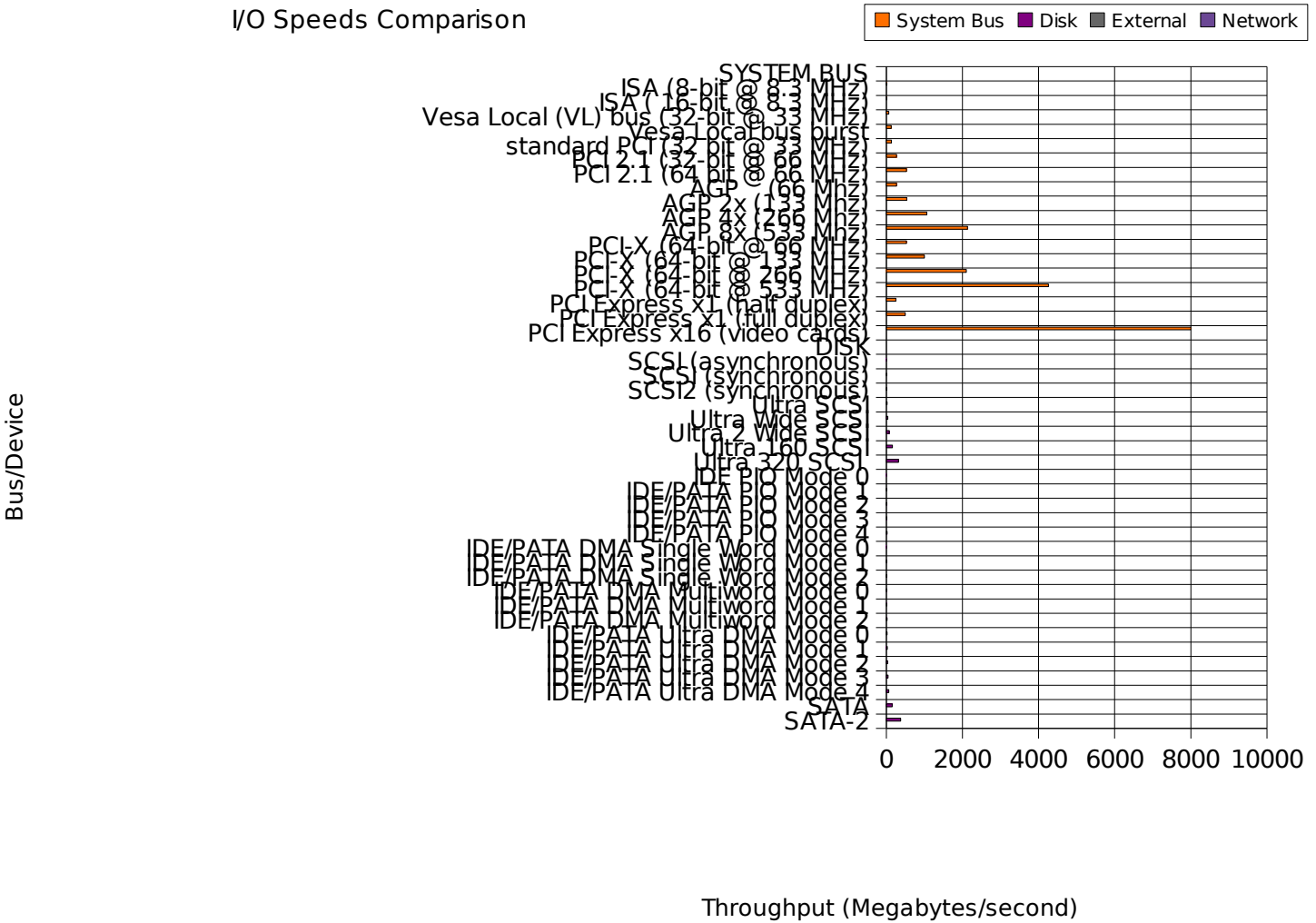
Hardware Performance - Memory

- More is generally better (but only if it's a bottleneck).
- 32-bit addressing limits
- BIOS settings
- Kernel settings
- Memory speed
- System bus speed
- ECC memory

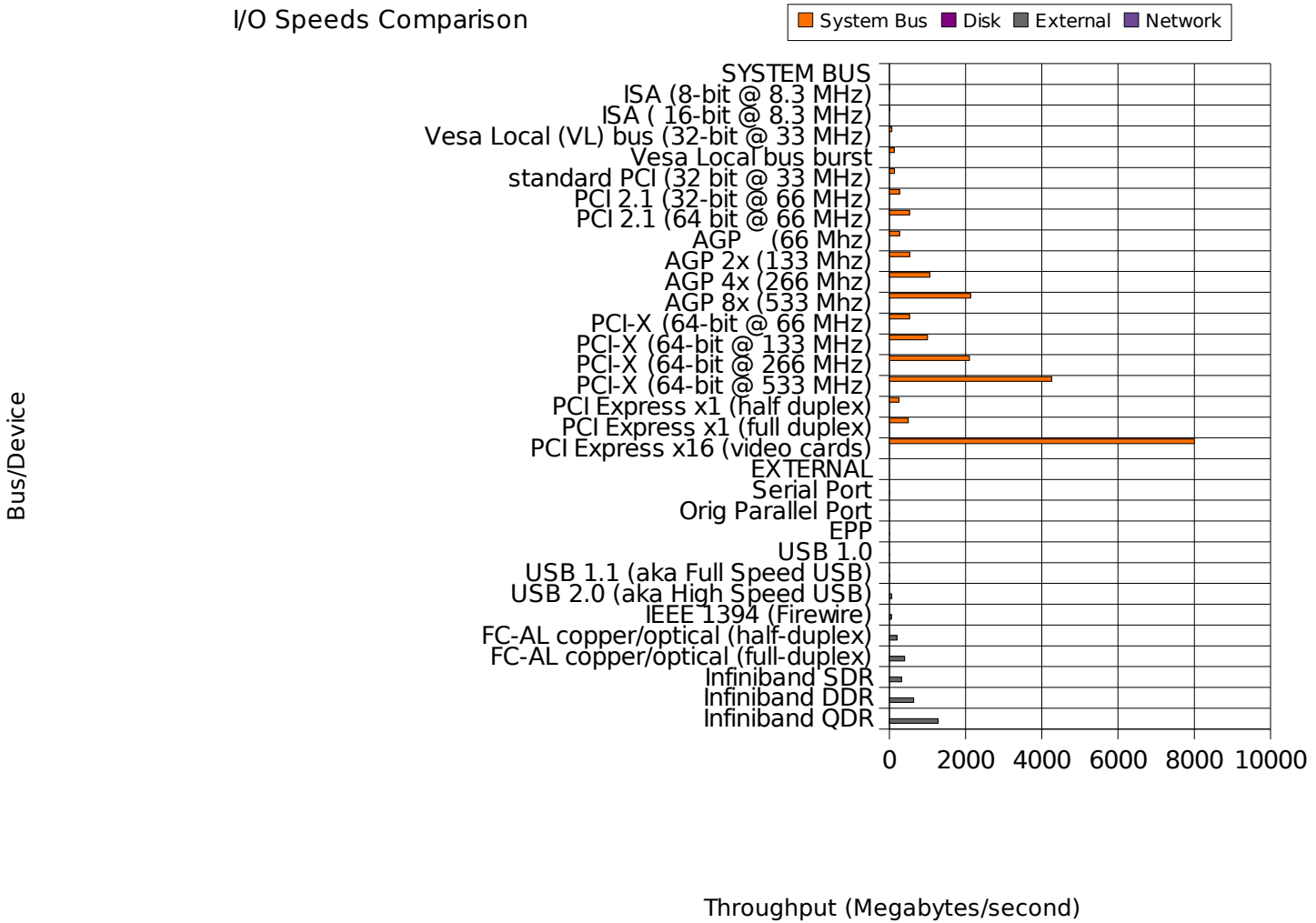
Hardware Performance – I/O (1/4)



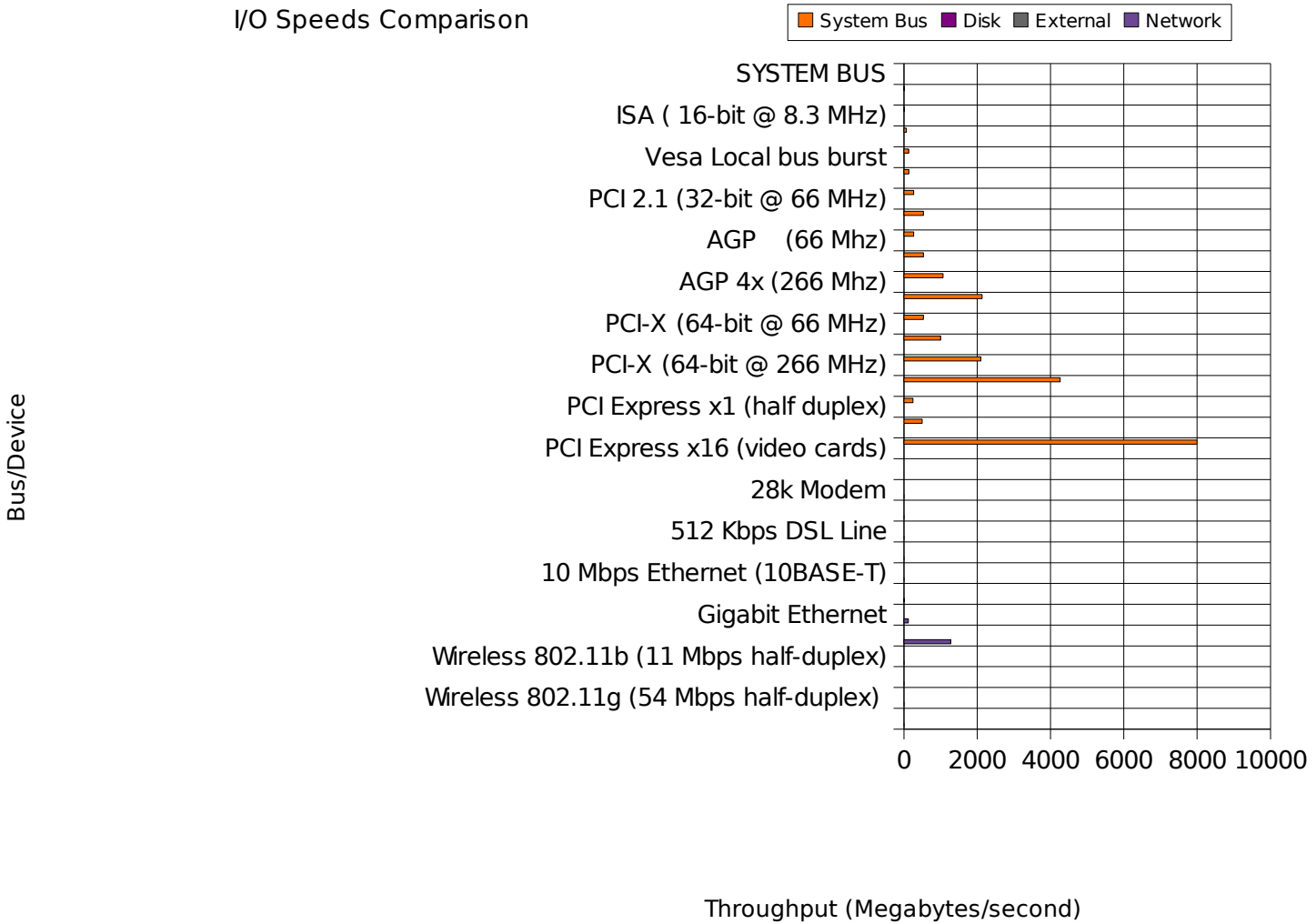
Hardware Performance – I/O (2/4)



Hardware Performance – I/O (3/4)



Hardware Performance – I/O (4/4)



Hardware Performance - Storage

- Hard drive performance characteristics
  - Storage capacity
  - Transfer rate
  - Average latency
  - Spindle speed
- Drive types
  - PATA / IDE
  - SATA
  - SCSI
- RAID
  - RAID 0
  - RAID 1
  - RAID 5
  - JBOD
  - RAID 0+1 or 1+0

Hardware Performance – Network (1/2)

- Ethernet
  - 10
  - 100 Fast
  - Gigabit
  - 10-Gigabit
- Processing Limits of Gigabit Ethernet
- High Performance Interconnects
  - Quadrics
  - Myrinet-2G
  - Myrinet-10G
  - Infiniband

Hardware Performance – Network (2/2)

- Jumbo Frames
  - larger MTU
  - less processing required
  - must be supported by all hardware
- Network Switch Backplanes
  - non-blocking
  - $2 \times n \times 1$ Gigabit
- Managed Switches
  - Monitoring
  - Bandwidth management
  - VLANs

OS Performance – Filesystem Tuning

- Match filesystem to workload
- Filesystem blocksize
  - 1024, 2048, 4096
  - number of inodes
- Separate different workloads
- Mount options
  - `noatime`
  - `nodiratime`
- Journalling parameters
  - logging mode
  - `barrier`
    - *ext3*
    - enable `barrier=1`
    - disable `barrier=0`
    - *ReiserFS*
    - enable `barrier=flush`
    - disable `barrier=none`

OS Performance – Filesystems

- ext2
  - standard linux fs
- ext3
  - journalling ext2
  - stable
- JFS
  - good support for large files, large directories
  - relatively new
- ReiserFS
  - particularly good for large numbers of small files
  - generally fast
- Reiser4
  - successor to ReiserFS
  - experimental
- XFS
  - low latency
  - suited to very large machines
  - slow on metadata creation and deletes

OS Performance – Other Filesystems

- Cluster filesystems
  - concurrent access shared storage
  - GFS
  - GFS2
  - OCFS
  - OCFS2
- Distributed filesystems
  - fault tolerant shared storage
  - client server model
  - NFS
  - Samba (CIFS)
  - Lustre

Performance Tuning Exercise 2

Suggest suitable filesystems for the following scenarios indicating what characteristics of that filesystem make it suitable,

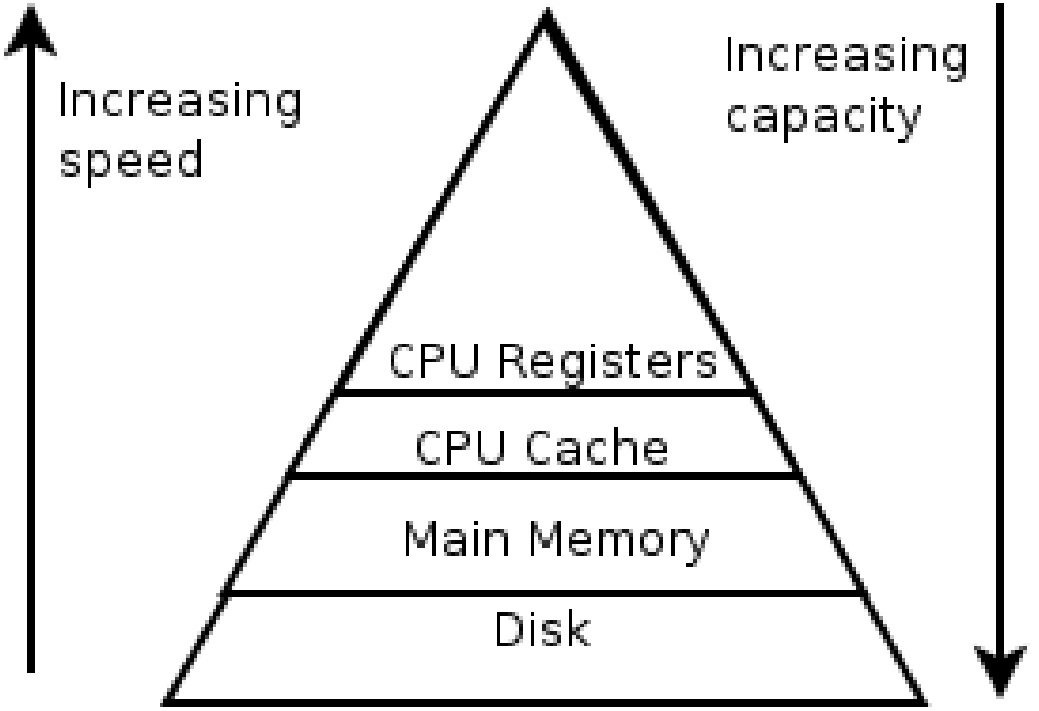
1) A Linux desktop used for word processing, web browsing and email.
2) A Samba fileserver serving a small workgroup.
3) A streaming media server.
4) An NNTP news server.
5) A mailserver for a medium sized company.
6) A 100-node HPC cluster.
7) An enterprise fileserver.

OS Performance – General

- Verify basic configuration
- Disable unneccesary services
  - `/etc/rcN.d`
- Remove unneccesary jobs
  - `cron`
- Restrict access to the system
  - development
  - test
  - production

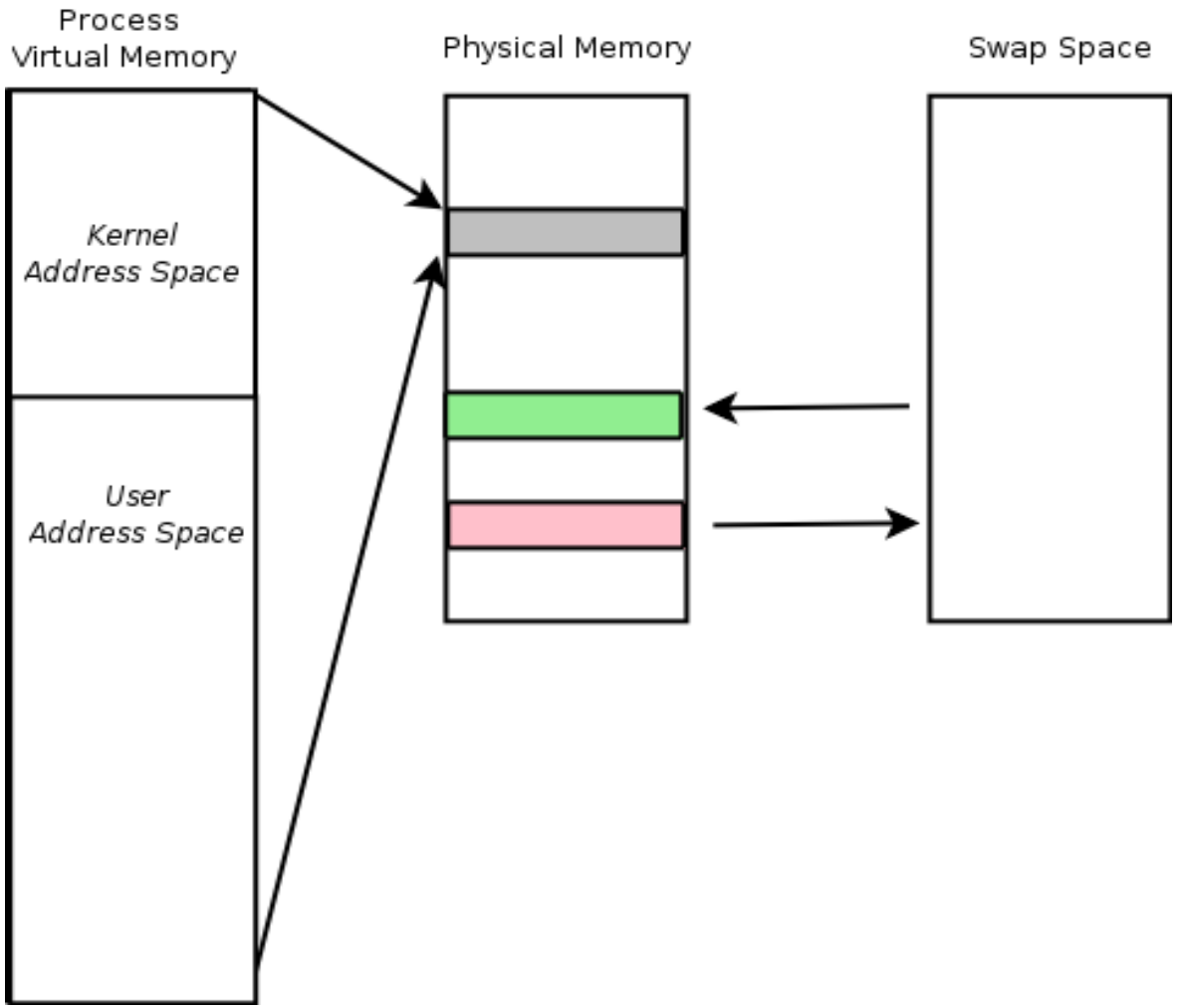
OS Performance – Virtual Memory (1/2)

- Sizing
  - memory required for application
- Monitoring
  - swap usage
  - % cpu spent paging
- Tuning
  - rate at which swap is used
  - cache sizes



OS Performance – Virtual Memory (2/2)

- Virtual Address space
  - 4 GB on 32-bit systems
  - 16 EB on 64-bit systems
  - Kernel space
  - User space
- Paging
- Performance
  - minimise paging



OS Performance – Drive tuning

- Tools
  - hdparm (IDE/PATA)
  - sdparm (SCSI)
- 2.6 I/O schedulers
  - cfq
  - deadline
  - as
  - noop
- 2.4 I/O scheduler
  - `elvtune`

OS Performance – Network Tuning – Core Settings

- Kernel Auto Tuning
- Socket Buffers
  - `net.core.wmem_default`
  - `net.core.rmem_default`
  - `net.core.rmem_max`
  - `net.core.wmem_max`
  - `net.core.netdev_max_backlog`
  - `net.core.somaxconn`
  - `optmem_max`

OS Performance – Network Tuning – TCP/IP Settings 1

- TCP buffer settings
  - `net.ipv4.tcp_rmem[]`
  - `net.ipv4.tcp_wmem[]`
  - `net.ipv4.tcp_mem[]`
- TCP Options
  - `net.ipv4.tcp_window_scaling`
  - `net.ipv4.tcp_sack`
- TCP connection Management
  - `net.ipv4.tcp_synack_retries`
  - `net.ipv4.tcp_retries2`

OS Performance – Network Tuning – TCP/IP Settings 2

- TCP Keep-Alive Management
  - `net.ipv4.tcp_keepalive_time`
  - `net.ipv4.tcp_keepalive_intvl`
  - `net.ipv4.tcp_keepalive_probes`
- IP Ports
  - `net.ipv4.ip_local_port_range`

OS Performance – CPU related tuning

- Linux 2.4 CPU Scheduler Characteristics
  - O(n)
  - Ok for servers
- Linux 2.6 CPU Scheduler Characteristics
  - O(1)
  - better SMP support
  - Good for both servers and desktops
- CPU Affinity
  - `taskset` command
- SMP IRQ Affinity
  - `/proc/irq/N/smp_affinity`

OS Performance – 2.4 Kernel tunables

- `/proc/sys/vm/bdflush`
  - kernel caches writes to disk in memory buffers
  - bdflush daemon periodically flushes these to disk
  - tune to suit characteristics of underlying disk
- `/proc/sys/vm/kswapd`
  - kernel pages/swaps data from memory to disk
  - kswapd manages this
  - tune to suit workload
- `/proc/sys/vm/max-readahead`
- `/proc/sys/vm/min-readahead`
  - control VFS prefetching
  - larger numbers increase memory consumption
  - may increase performance for linear file reads

OS Performance – 2.6 Kernel tunables (1/2)

- /proc/sys/fs/file-max
- /proc/sys/fs/inode-max
- /proc/sys/vm/overcommit_memory
- /proc/sys/vm/overcommit_ratio
- /proc/sys/vm/dirty_ratio
- /proc/sys/vm/dirty_background_ratio
- /proc/sys/vm/dirty_expire_centisecs
- /proc/sys/vm/dirty_writeback_centisecs

OS Performance – 2.6 Kernel tunables (2/2)

- `/proc/sys/vm/swapiness`

Performance Tuning Exercise 3

Suggest some tuning that could be performed for the following scenarios,

1) A database server.
2) A Java application server.
3) A desktop system.
4) A batch processing system.

Performance Monitoring – CPU Utilisation

- uptime
- `/proc/cpuinfo`
- vmstat
- top

Performance Monitoring – Memory Utilisation

- `free`
- `/proc/meminfo`
- `/proc/slabinfo`
- `ps`
- `vmstat`

Atlantic LINUX

Performance Monitoring – I/O Utilisation

- `vmstat`
- `iostat`

Performance Monitoring – `sar`

- Collect
  - I/O and transfer rate statistics
  - Paging statistics
  - Process creation counts
  - Block device activity
  - Network statistics
  - Processor activity
  - Queue lengths
  - Memory and space space utilisation
- Reports
- Trending

Performance Monitoring – network

- Managed switch
  - diagnostics
  - port mirroring
- Commands
  - `ifconfig`
  - `netstat`
  - `tcpdump`
  - `wireshark`
  - `iptraf`

Performance Tuning – Web Servers

- Use Apache 2.x rather than Apache 1.x.
- Ensure plenty of RAM.
- Ensure webserver is never swapping.
- Tune application (MaxClients setting).
- Consider load balancing multiple servers.
- Split dynamic and static load to different servers.

Performance Tuning – File & Print Server

- Use fast storage hardware
- Use software RAID
- Use a fast filesystem (ReiserFS or XFS)
- Mount filesystems with *noatime* option
- Consider using the *cfq* or *deadline* I/O scheduler
- Keep reads separate from random writes
- Lots of RAM

Performance Tuning – Database Servers

- Lots of RAM
- 64-bit processor
- Consider using the *deadline* I/O scheduler
- Tune the page cache down (but not too much)
- Tune shared memory
- Increase network buffers
- Spread database load over multiple disks

Performance Tuning – Java App Server

- Tune JVM memory settings
- Ensure sufficient network bandwidth
- Buy fast CPUs
- Buy lots of RAM
- Benchmarks!

Tuning C (1/2)

- Problems
  - larger code
  - debugging difficulty
  - errors
  - compile time
- Optimisation levels
  - `O0`
  - `O1`
  - `O2`
  - `O3`
  - `Os`
- Profiling

Tuning C (2/2)

- Vectorisation
- Architecture-specific code
  - `-march=`*`cpu`*
  - `-mtune=`*`cpu`*
- Other Compilers
  - Intel
    - C, C++
    - Fortran (F77, F95)
  - The Portland Group
    - C, C++
    - Fortran (F77, F95)

Tuning Java (1/2)

- General
  - Only performance tune bottleneck code
  - Use the latest JDK
  - Use `StringBuffer` for string concatenation
  - Use `log.isDebugEnabled()`
  - Profile, profile, profile
- JDBC
  - `PreparedStatement`
  - Placeholders
  - Tune the SQL
  - Connection pooling
  - Stored procedures

Tuning Java (2/2)

- JVM Settings
  - many options
  - heap size
  - garbage collection algorithm
  - `-server`
  - `-client`
- Microbenchmarking gotchas

```
java -server -Xmx512m -Xms512m -XX:+UseConcMarkSweepGC
```

Application Profiling

- `time`
  - user time and kernel time breakdown
- `top`
  - basic realtime resource usage
- `strace`
  - system call trace
- `gprof`
  - user call graphs
- `valgrind`
  - call graphs
  - memory usage analysis
- `oprofile`
  - system-wide profiling
  - low overhead

Performance Tuning Exercise 4

1) Write a program in Java or C to write 1 million X's to a file. Using some of the lessons learned in the course, test the program and tune it for performance. The following is a sample starting point in Java,

```
import java.io.FileOutputStream;
public class SpeedTest
{
  public static void main (String[]args) throws Exception
  {
    String filename = args[0];
    FileOutputStream fout = new FileOutputStream (filename);
    for (int i = 0; i < 1000000; i++)
    {
        fout.write ('X');
    }
    fout.close ();
  }
}
```

In closing ...

- Summary
- Next steps
- Questionaire

Thank you and well done!