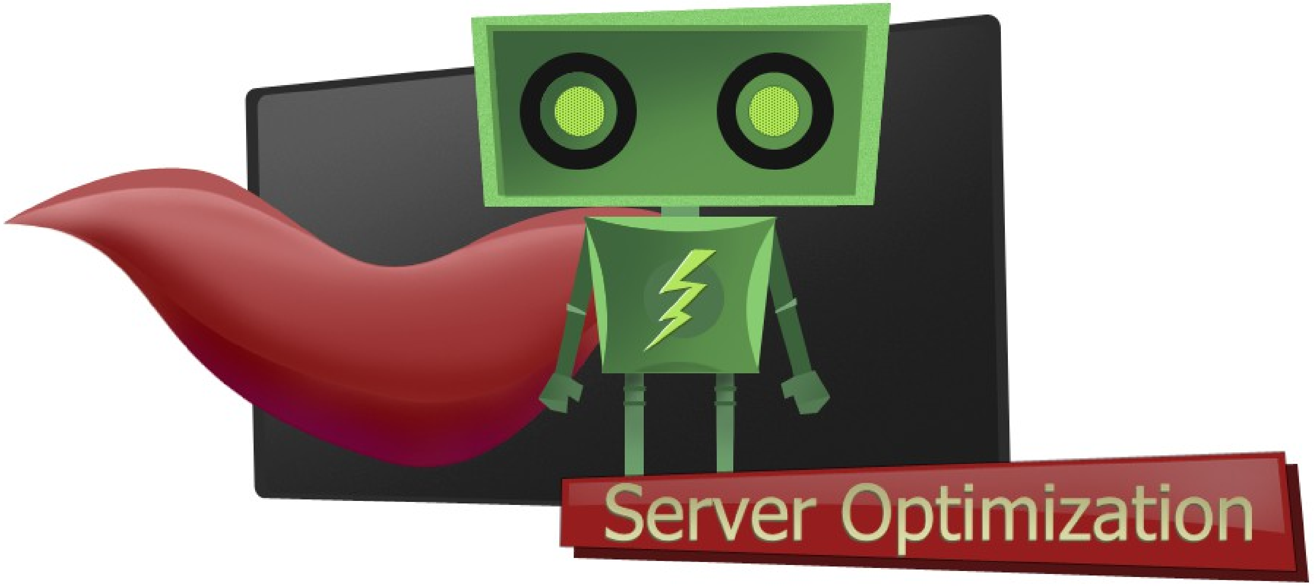
Optimizing Linux Servers



Davor Guttierrez
dguttierrez@me.com
3 Gen d.o.o.

Davor Guttierrez

3 Gen d.o.o

since 1996

located in Ljubljana; Slovenia

33 employees

large IBM mainframe systems and

UNIX-oriented systems along with databases

more on www.3Gen.si or facebook.com/3GenSI

Agenda

- What is optimization?
- Performance (software / hardware)
- Server optimization
- Performance monitoring
- System Monitoring Tools
- Benchmarking Tools

What is optimization?

- Our server is slow
  - we have new very expensive server but ...
  - we have new Linux distribution but ...

- What is slow in your server?
  - too many services running, ...
  - disk, I/O, network configuration, ...

Performance

To boost performance of a server, we need just both its hardware and software components to make it operate efficiently.

Server optimization

Optimization can include fine tuning of

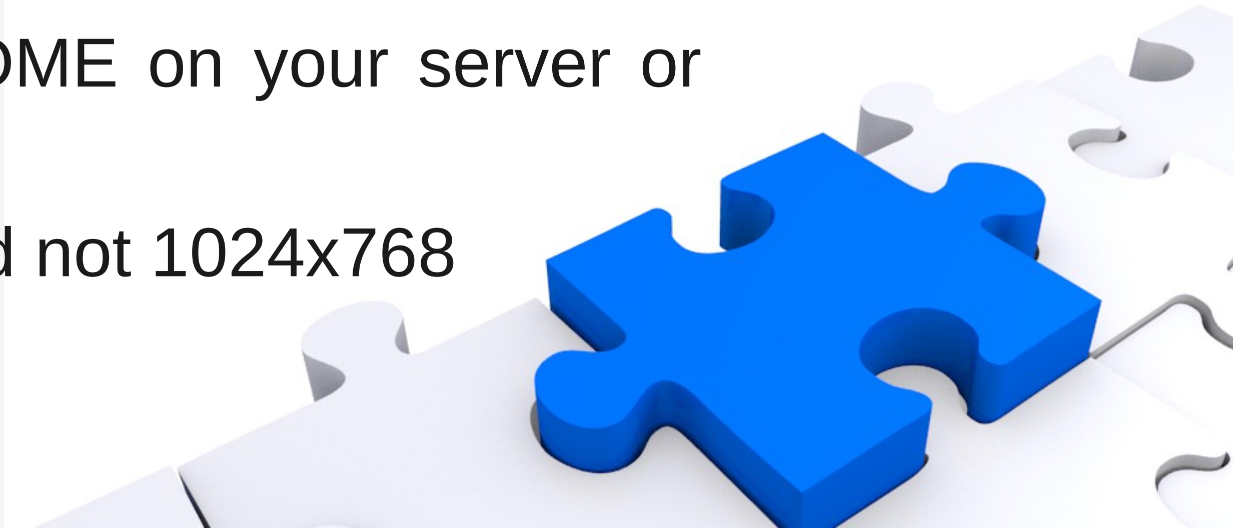
- web servers (Apache, lighttpd, nginx etc),
- disk I/O, block devices, RAID or different filesystems (including SCSI and SSD devices),
- kernel,
- network I/O, TCP/IP network stack,
- firewall etc.

as well as

- databases optimization – benchmarking and profiling, finding bottlenecks, settings optimization;
- data storages tuning;
- disk and memory usage optimization

Start here … with installation

- Always make custom installation of server, don't use default settings and default install
- Custom partitioning,multiple file systems, ...
- Install only needed packages, never install whole group of packages – make minimal installation and then add packages
- You don't need X Window and GNOME on your server or you do?
- Console is 80x25 characters long and not 1024x768



Performance monitoring

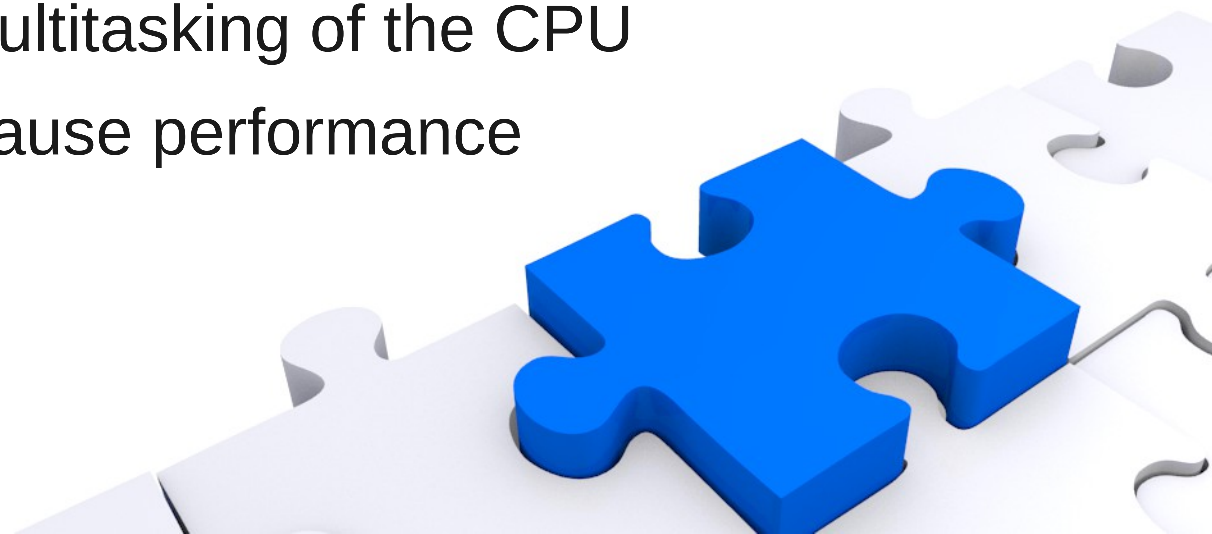
- Linux system administrators should be proficient in Linux performance monitoring and tuning.To identify system bottlenecks and come up with solutions to fix it, you should understand how various components of Linux works.
- On a very high level, following are the four subsystems that needs to be monitored:
  - CPU
  - Network
  - I/O
  - Memory

CPU

- You should understand the four critical performance metrics for CPU
  - context switch,
  - run queue,
  - cpu utilization,
  - load average.

Context Switch

- When CPU switches from one process (or thread) to another, it is called as context switch
- When a process switch happens, kernel stores the current state of the CPU (of a process or thread) in the memory
- Kernel also retrieves the previously stored state (of a process or thread) from the memory and puts it in the CPU
- Context switching is very essential for multitasking of the CPU
- A higher level of context switching can cause performance issues



Run queue

- Run queue indicates the total number of active processes in the current queue for CPU
- When CPU is ready to execute a process, it picks it up from the run queue based on the priority of the process
- Processes that are in sleep state, or I/O wait state are not in the run queue
- A higher number of processes in the run queue can cause performance issues

CPU Utilization

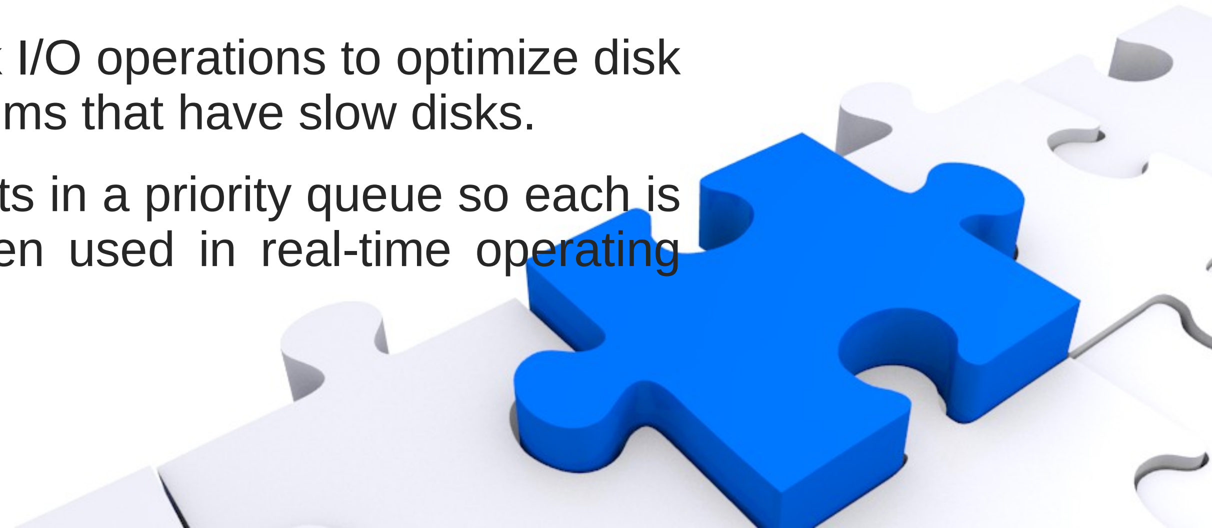
- This indicates how much of the CPU is currently getting used
- This is fairly straight forward, and you can view the CPU utilization from the top command
- 100% CPU utilization means the system is fully loaded
- A higher % of CPU utilization will cause performance issues

LOAD average

- This indicates the average CPU load over a specific time period.
- On Linux, load average is displayed for the last 1 minute, 5 minutes, and 15 minutes. This is helpful to see whether the overall load on the system is going up or down.
- Load average of “0.25 1.20 1.90” indicates that the load on the system is coming down. 0.25 is the load average in the last 1 minute. 1.20 is the load average in the last 5 minutes. 1.90 is the load average in the last 15 minutes.
- This load average is calculated by combining both the total number of process in the queue, and the total number of processes in the uninterruptable task status.

Disk I/O optimization

- Linux currently ships with four different I/O schedulers. They are: deadline, noop, anticipatory, and cfq. There are many differences between these scheduling algorithms:
- CFQ: This is the default algorithm in most Linux distributions. It attempts to distribute all I/O bandwidth evenly among all processes requesting I/O. It is ideal for most purposes.
- NOOP: The noop algorithm attempts to use as little cpu as possible. It acts as a basic FIFO queue expecting the hardware controller to handle the performance operations of the requests.
- Anticipatory: This algorithm attempts to reorder all disk I/O operations to optimize disk seeks. It is designed to increase performance on systems that have slow disks.
- Deadline: This scheduling algorithm places I/O requests in a priority queue so each is guaranteed to be ran within a certain time. It is often used in real-time operating systems.



System scheduler … how to change

-
  ```
  cat /sys/block/sda/queue/scheduler
  ```

  change it with:

-
  ```
  echo noop > /sys/block/sda/queue/scheduler
  ```

  in SuSE Linux via YaST ...

More about System scheduler

- Changing schedulers on the fly allows you to test and benchmark the algorithms for your specific application
- Once the change is issued, any current I/O operations will be executed before the new scheduler goes into effect, so the change will not be instant
- Also remember that once one is set and performs to your liking, be sure to set the change to be applied on subsequent reboots

And a little bit more ...

- It is often recommend to use noop or deadline on any SSD drive
- There is usually no definitive answer to which algorithm to use
- Benchmarking each one will be your best option
- There are cases where cfq may not be the best scheduler for your system or application
- An example is if you are running a RAID disk array with a caching raid controller

I/O optimization

- I/O wait is the amount of time CPU is waiting for I/O. If you see consistent high i/o wait on you system, it indicates a problem in the disk subsystem.
- You should also monitor reads/second, and writes/second. This is measured in blocks. i.e number of blocks read/write per second. These are also referred as bi and bo (block in and block out).
- tps indicates total transactions per seconds, which is sum of rtps (read transactions per second) and wtps (write transactions per seconds).

Disk I/O optimization

- Filesystem to use:
  - EXT2
  - EXT3
  - ReiserFS
  - EXT4
  - BTRFS

Use FS options in fstab (noatime, ...)

Disk I/O optimization

- RAID (software or hardware)
  - RAID0
  - RAID1
  - RAID5 (not recomended)
  - RAID10

Disk optimization

- Use benchmark programs like Bonnie++
- Use hdparm
- Upgrade BIOS of your server and Firmware of your disk
- Does your disk park heads?

Network Tuning

- A good understanding of TCP/IP concepts is helpful while analyzing any network issues.
- For network interfaces, you should monitor total number of packets (and bytes) received/sent through the interface, number of packets dropped, etc.,

TCP Tuning

For servers that are serving up huge numbers of concurent sessions, there are some tcp options that should probabaly be enabled. With a large # of clients doing their best to kill the server, its probabaly not uncommon for the server to have 20000 or more open sockets.

Allows more local ports to be available

```
echo 1024 65000 > /proc/sys/net/ipv4/ip_local_port_range
```

Increasing the amount of memory associated with socket buffers can often improve performance

```
echo 262143 > /proc/sys/net/core/rmem_max

echo 262143 > /proc/sys/net/core/rmem_default
```

These reduce the amount of work the TCP stack has to do, so is often helpful in this situation

```
echo 0 > /proc/sys/net/ipv4/tcp_sack

echo 0 > /proc/sys/net/ipv4/tcp_timestamps
```

TCP Tuning (10GB NIC)

```
# increase TCP max buffer size settable using setsockopt()
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
# increase Linux autotuning TCP buffer limit
net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216
# increase the length of the processor input queue
net.core.netdev_max_backlog = 30000
# recommended default congestion control is htcp
net.ipv4.tcp_congestion_control=htcp
# recommended for hosts with jumbo frames enabled
net.ipv4.tcp_mtu_probing=1
```

TCP Congestion Avoidance Algorithms

- reno: Traditional TCP used by almost all other operating systems. (default)
- cubic: CUBIC-TCP
- bic: BIC-TCP
- htcp: Hamilton TCP
- vegas: TCP Vegas
- westwood: optimized for lossy networks

Which do u use?

```
davorg@kranch:~$ sudo sysctl net.ipv4.tcp_available_congestion_control

net.ipv4.tcp_available_congestion_control = cubic reno
```

Memory Optimization

- If you have 16 GB RAM installed on your system, you have 16 GB of physical memory
- Virtual memory = Swap space available on the disk + Physical memory. The virtual memory contains both user space and kernel space
- Using either 32-bit or 64-bit system makes a big difference in how much memory a process can utilize
- On a 32-bit system a process can only access a maximum of 4GB virtual memory
- On a 64-bit system there is no such limitation

More about Memory optimization ...

- Unused RAM will be used as file system cache by the kernel
- Linux system will swap when it needs more memory. i.e when it needs more memory than the physical memory
- When it swaps, it writes the least used memory pages from the physical memory to the swap space on the disk
- Lot of swapping can cause performance issues, as the disk is much slower than the physical memory, and it takes time to swap the memory pages from RAM to disk

Memory optimization

- Dense Memory (Hardware specific)
- NUMA (Non Uniform Memory Access)
- Huge Pages
- Manage Virtual Memory pages
  - Flushing of dirty pages
  - Swapping behavior

NUMAD

- User-level daemon to automatically improve out of the box NUMA system performance
- Fedora 17
- RHEL 6.3 as tech preview
- Not enabled by default

- Monitors available system resources on a per-node basis and assigns significant consumer processes to aligned resources for optimum NUMA performance.
- Rebalances when necessary
- Provides pre-placement advice for the best initial process placement and resource affinity.

Huge Pages

- 2M pages vs 4K standard linux page
- Virtual to physical page map is 512 times smaller
- TLB can map more physical pages, resulting in fewer misses
- Traditional Huge Pages always pinned
- Transparent Huge Pages in RHEL6
- Most databases support Huge pages
- 1G pages supported on newer hardware

How to configure Huge Pages (16G)

```
echo 8192 > /proc/sys/vm/nr_hugepages
vi /etc/sysctl.conf (vm.nr_hugepages=8192)
```

Flushing Caches

- Drop unused Cache
- Frees unused memory
- File cache
- If the DB uses cache, may notice slowdown

- Free pagecache

```
echo 1 > /proc/sys/vm/drop_caches
```

- Free slabcache

```
echo 2 > /proc/sys/vm/drop_caches
```

- Free pagecache and slabcache

```
echo 3 > /proc/sys/vm/drop_caches
```

Swappiness

- Controls how aggressively the system reclaims “mapped” memory:
- Default - 60%
- Decreasing: more aggressive reclaiming of unmapped pagecache memory
- Increasing: more aggressive swapping of mapped memory

80/20

- **Remember the 80/20 rule**

  80% of the performance improvement comes from tuning the application, and the rest 20% comes from tuning the infrastructure components.

System Monitoring Tools

- vmstat
- netstat
- ps
- top
- atop, htop
- mtop
- Iostat
- xosview

Kernel Tuning

- Recompile your kernel
- Exclude unneeded modules
- Use RealTime kernel
- Make kernel smaller

Samba tuning

- enable AIO (EXTREME)
- socket options = TCP_NODELAY IPTOS_LOWDELAY SO_RCVBUF=65536 SO_SNDBUF=65536
- raw read and write
- oportunistic lokcing
- log level

Database optimization

- MySQL – use MySQL tuning App
- Optimize table
- Do you use InnoDB or MyISSAM DB?
- Change database if possible

OpenLDAP Tuning

- The most important tuning aspect for OpenLDAP is deciding what attributes you want to build indexes on.

```
Cachesize 10000
dbcachesize 100000
sizelimit 10000
loglevel 0
dbcacheNoWsync

index cn,uid
index uidnumber
index gid
index gidnumber
index mail
```

- If you add the following parameters to /etc/openldap/slapd.conf before entering the info into the database, they will all get indexed and performance will increase.

Apache Tuning

Make sure you starting a ton of initial daemons if you want good benchmark scores.

Something like:

```
MinSpareServers 20
MaxSpareServers 80
StartServers 32
```

this can be higher if apache is recompiled

```
MaxClients 256
MaxRequestsPerChild 10000
```

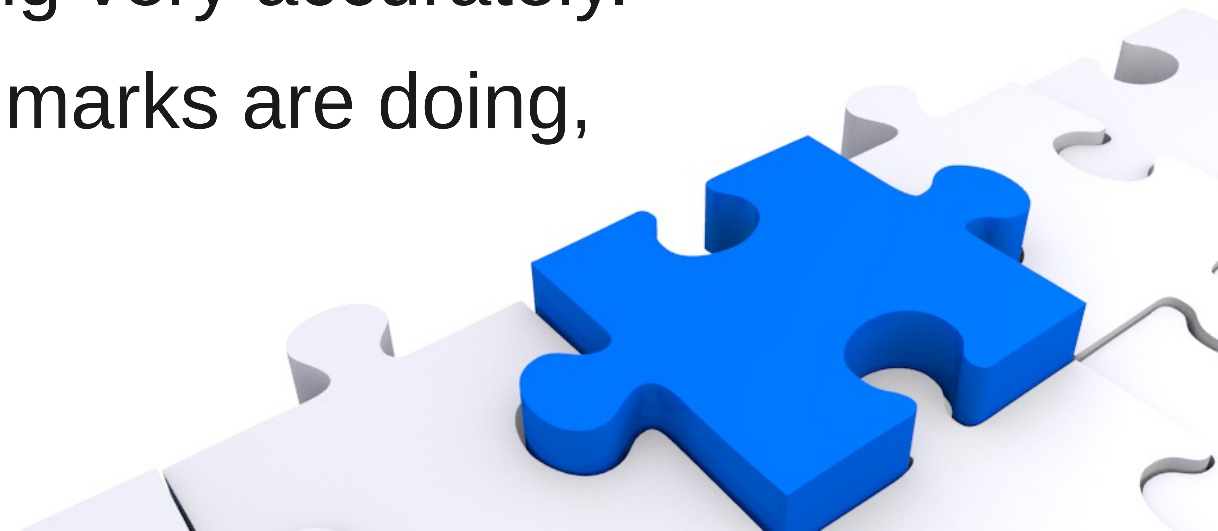
Note: Starting a massive amount of httpd processes is really a benchmark hack. In most real world cases, setting a high number for max servers, and a sane spare server setting will be more than adequate. It's just the instant on load that benchmarks typically generate that the StartServers helps with.

Slow websites

- Use optimizers
- Use memcache
- Tune your apache
- Minimize number of Apache modules
- Change Apache for Nginx

Benchmark

- A good set of benchmarking utilities are often very helpful in doing system tuning work. It is impossible to duplicate "real world" situations, but that isnt really the goal of a good benchmark.
- A good benchmark typically tries to measure the performance of one particular thing very accurately.
- If you understand what the benchmarks are doing, they can be very useful tools.



Benchmark Tools

- **bonnie++** - is a free file system benchmarking tool for Unix-like operating systems
- **DBench** - is a tool to generate I/O workloads to either a filesystem or to a networked CIFS or NFS server
- **http_load** - runs multiple http fetches in parallel, to test the throughput of a web server
- **dkftpbench** - measuring how many simultaneous dialup users can be down loading from an FTP site at the same time
- **tiobench** - is a multi-threaded I/O benchmark
- **ttcp** - is a utility program for measuring network throughput
- **netperf** – network performance tester

Identify and solve performace issue

- Understand the problem
  - Half of the problem is solved when you clearly understand what the problem is.
- Monitor and collect data
  - After defining the problem clearly, monitor the system and try to collect as much data as possible on various subsystems
- Eliminate and narrow down issues
  - After having a list of potential issues, dive into each one of them and eliminate any non issues
- Make one change at a time
  - don’t try to make multiple changes at one time

TNX ...

- E-mail: dguttierrez@me.com
- Company: www.3Gen.si
- Blog: www.d-mashina.net
- CV: www.guttierrez.org