

Work with InterSystems.

Not separate systems.



Caché Performance Measurement and Tuning

Neil Alton

WRC Performance Team

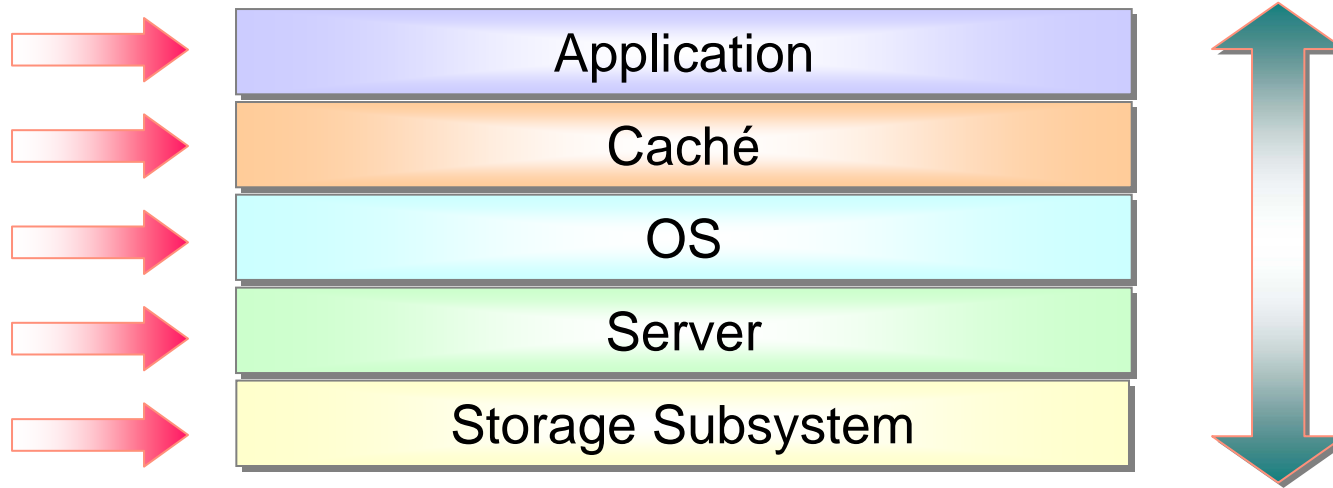
INTERSYSTEMS

Agenda



- Brief performance review
- Performance tools in Caché
- Performance components
- Some real world examples
- Questions

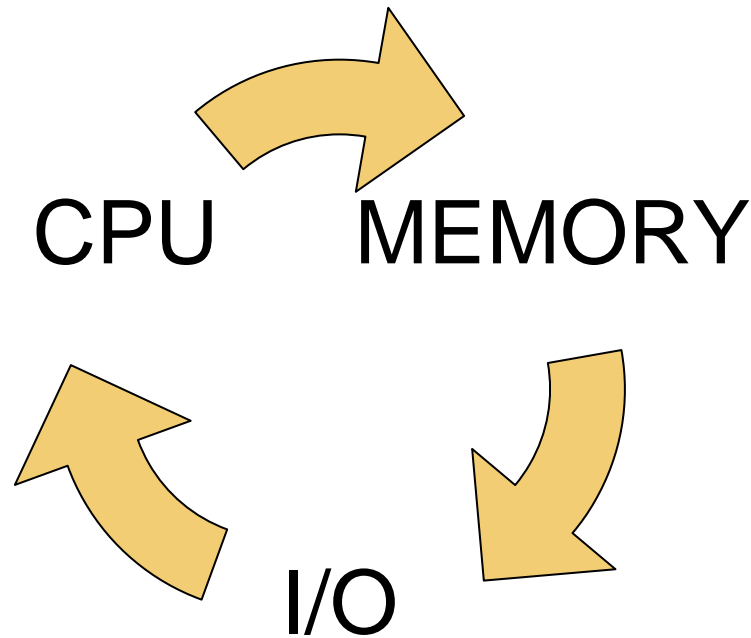
Performance review (i)



Performance review (ii)



Performance is a cycle



Performance tools provided



- SMP
- %Monitor
- PERFMON
- %SYS.MONLBL

- OS provided tools
(sar, vmstat, T4, MS Perfmon)

- pButtons

pButtons – what is it?



- Similar idea to ^Buttons
- Specifically for performance data – so runs over time
- 2 steps – “run” and “collect”
- Uses temporary log files
- Data gathered in parallel, so **not** a single job
- Gathers configuration/setup data as well as real time data
- In %SYS namespace
- Collect step creates a consolidated html file

pButtons - profiles



- Daily monitoring or focused investigation?
- Customize to suit demand (some presets provided)
- A “profile” defines
 - Sample duration (seconds)
 - Number of samples taken
- Profiles can run in parallel
- API to add and edit profile definitions

pButtons – global driven & API



- When run, creates a global containing
 - monitoring commands
 - profile definitions
- Template but customized by you!
- Run() and Collect() calls
 - easy integration into Task Manager
- Collect automatically run on profile start
- Collect leaves running profiles alone

pButtons - examples



Quick demo

Scalability



- Development single user vs deployment
 - Understand what your application does and needs
 - Will ECP be used in deployment? Things to look out for
- Predictability
 - What resources are needed to run this code with X users?
 - Will running this code in a concurrent user environment create contention otherwise not seen in a single user development cycle

CPUs – is more GHz beneficial?

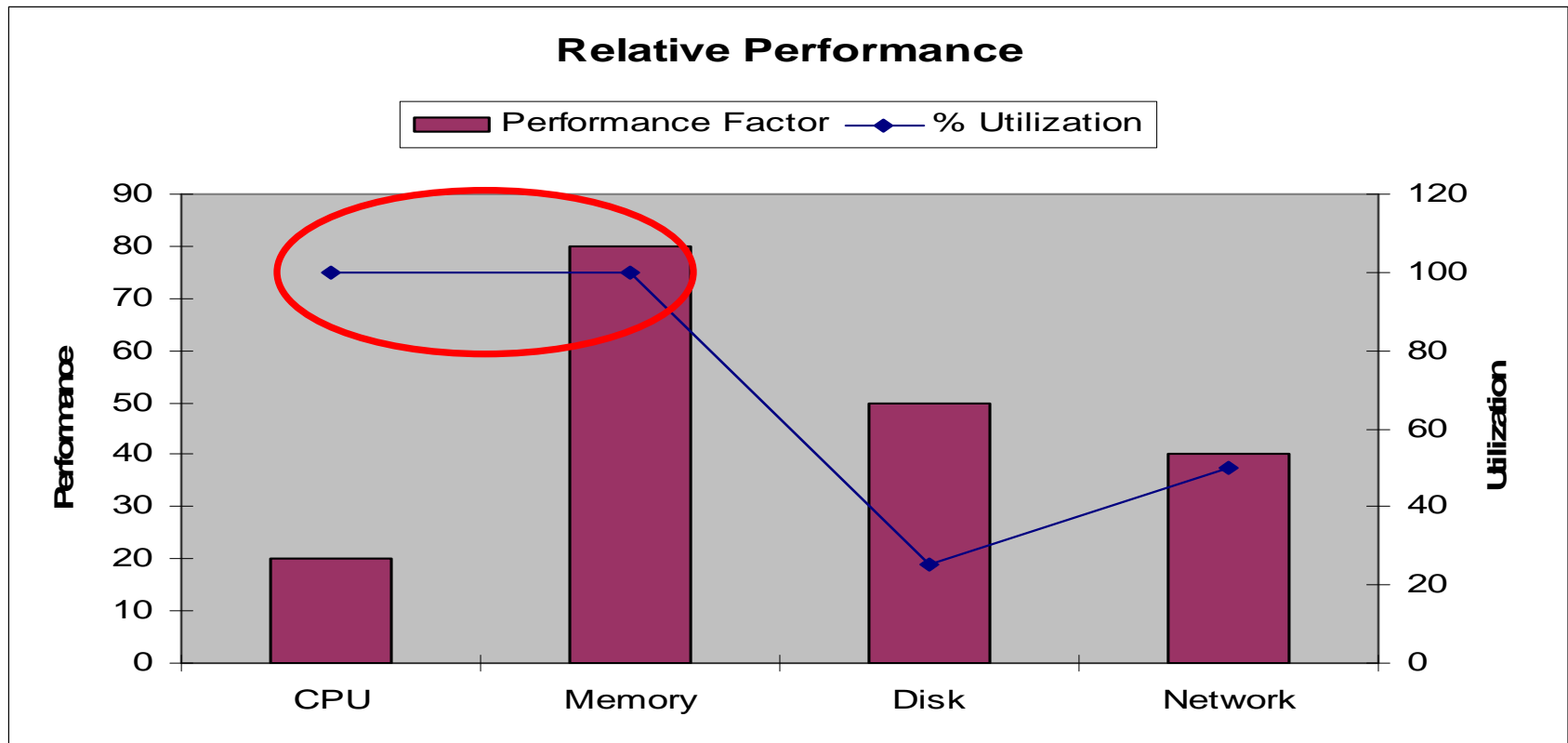


- The short answer is YES – However...
- Concurrency and timeslice
- Impact on other performance factors
(Disk, Network, Memory)
- 64-bit does not necessarily mean “FASTER”

CPUs – is more better (i)?



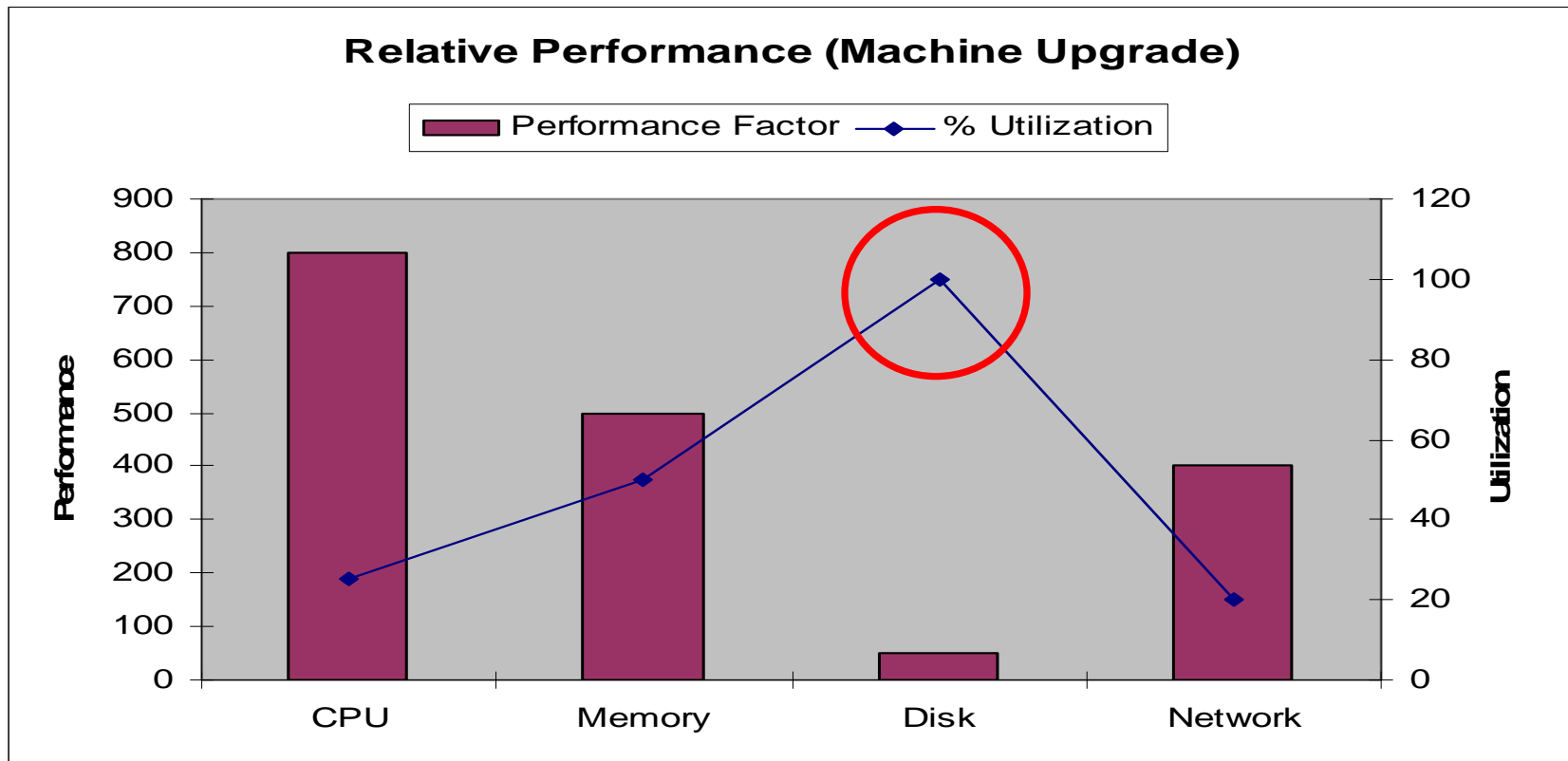
- Current performance shows CPU and Memory bottleneck



CPUs – is more better (ii)?



- CPU and Memory Upgrade does not solve all issues



Memory – effect of large pages (Windows, Linux, and AIX)



- Large pages improve memory access times
 - this increases the efficiency of the translation buffer, which can increase performance for frequently accessed memory
- The key is having these pages locked into memory
- Memory allocation “retries” on Unix, Linux, and VMS
 - retries limited times with a series of requests decreasing the amount of space requested with each iteration
- Memory allocation in 2 phases on Windows
 - Request to use large pages
 - Request size to allocate

Reference Slide

Allocating Linux Large Pages (1)



Cache automatically uses Huge Pages on Linux if the operating system has been so configured.

Configuring Huge Pages

The size of the Huge Pages pool is specified by the desired number of Huge Pages. To calculate the number of Huge Pages you first need to know the Huge Page size. To obtain the size of Huge Pages, execute the following command:

```
$ grep Hugepagesize /proc/meminfo
Hugepagesize: 2048 kB
$
```

The output shows that the size of a Huge Page on my system is 2MB. This means if I want to allocate a 1GB Huge Pages pool, then I have to allocate 512 Huge Pages. The number of Huge Pages can be configured and activated by setting `nr_hugepages` in the `proc` filesystem. For example, to allocate 512 Huge Pages, execute:

```
# echo 512 > /proc/sys/vm/nr_hugepages
```

Alternatively, you can use `sysctl(8)` to change it:

```
# sysctl -w vm.nr_hugepages=512
```

To make the change permanent, add the following line to the file `/etc/sysctl.conf`. This file is used during the boot process. The Huge Pages pool is usually guaranteed if requested at boot time:

```
# echo "vm.nr_hugepages=512" >> /etc/sysctl.conf
```

If you allocate a large number of Huge Pages, the execution of the above commands can take a while. To verify whether the kernel was able to allocate the requested number of Huge Pages, run:

```
$ grep HugePages_Total /proc/meminfo
HugePages_Total: 512
```

Reference Slide

Allocating Linux Large Pages (2)



If you allocate a large number of Huge Pages, the execution of the above commands can take a while. To verify whether the kernel was able to allocate the requested number of Huge Pages, run:

```
$ grep HugePages_Total /proc/meminfo
HugePages_Total: 512
$
```

The output shows that 512 Huge Pages have been allocated. Since the size of Huge Pages is 2048 KB, a Huge Page pool of 1GB has been allocated and pinned in physical memory.

For example, suppose a 64 page pool (132MB) has been permanently allocated in the boot routine. Configure your cache instance to request 32MB global buffers. The result will be approximately 60MB shared memory requested.

Check the Huge Page pool before startup:
>cat /proc/meminfo

```
...
HugePages_Total: 64
HugePages_Free: 64
```

```
Hugepagesize: 2048 kB
```

After startup, check the pool again:
>cat /proc/meminfo

```
...
HugePages_Total: 64
HugePages_Free: 35
```

```
Hugepagesize: 2048 kB
```

Reference Slide

Allocating AIX Large Pages (1)



```
root@p57062:/home/isc/c810/mgr# vmo -r -o lgpg_size=16777216 -o lgpg_regions=350
```

Setting lgpg_size to 16777216 in nextboot file

Setting lgpg_regions to 350 in nextboot file

Warning: although the change concerns a dynamic tunable, bosboot should be run to optimally restore these settings at reboot

Warning: changes will take effect only at next reboot

This command sets the Large Page size to 16MB and allocates 350 such pages (regions) for a total amount of about 5GB. In order to complete the setting, bosboot command needs to be run and then the system rebooted:

```
root@p57062:/# bosboot -a
```

bosboot: Boot image is 36562 512 byte blocks.

After the reboot, Large Pages in shared segments need to be pinned in memory:

```
vmo -o v_pinshm=1
```

2) checking Large Pages allocation

vmo -L command will show that the values have been changed:

Reference Slide

Allocating AIX Large Pages (2)



```
root@p57062:/# vmo -L
```

```
NAME          CUR  DEF  BOOT  MIN  MAX  UNIT  TYPE  DEPENDENCIES...
-----
lgpg_regions   0   0   350   0   8E-1      D  lgpg_size
-----
lgpg_size      0   0   16M   0   16M  bytes  D  lgpg_regions
-----
```

The above output is produced after setting the values. Upon reboot, the column CUR will also show the changed values. However, there is no notification to Cache' whether the Large Page allocation has succeeded or not. To verify that Cache' is indeed using Large Pages, find a Cache' process and run the following command:

```
root@p57062:/# svmon -P 82928
```

```
-----
Pid Command      Inuse   Pin   Pgspace Virtual 64-bit Mthrd 16MB
82928 cache      1200166 1193952 0 1199312  Y  N  Y
  PageSize      Inuse   Pin   Pgspace Virtual
  s  4 KB        2982   0     0     2128
  m  64 KB       584   382   0     584
  L  16 MB       290   290   0     290
```

You can see that 290 16MB pages are used here.

Buffering conflicts



- By default the OS will buffer all I/O
- Cache has data in global buffers
- => double buffering – waste of memory!

I/O – is a single huge disk good?



- Depends on what you call a “single” disk...
- A single spindle – know the limitations – do the maths
- Write Daemon assignments
 - New alternate Write Daemon strategy
- When large disks are good

I/O – SAN best practices



- Engage the storage vendor to inform them of the IO patterns of your application
- Understanding what RAID Levels are appropriate
- Controller cache – more is ALWAYS better
- Physical spindles – more is ALWAYS better...BUT setting it up right is important
- Just because the SAN is *expensive* doesn't mean it was configured correctly



- ECP
 - Do you have ample bandwidth?
 - What kind of latency do you have?
- Shadowing
 - Do you have ample bandwidth?
 - What kind of latency do you have?
 - Is TCP setup right on both sides?



- New in 2008.2 CSP Gateway
- Performance logging (to CSP.log) at request level, for:
 - Total time to service request
 - Obtain a [new] connection to Caché
 - Send request to Caché
 - Receive response from Caché
 - Caveat – disk intensive, so not really for production

What is my application doing?



- Didn't you write it in the first place?
- It should be faster than this.
- What is different now?
- What changed? !!!
- Why isn't it scaling?
- Where is the bottleneck?
- Are you sure there's only one!
- What is consuming all my resources?

Now some real world examples

Example 1



- Write latency on disk greater than 8ms because of synchronous replication to remote datacenter
- Using Cache' clusters and performance was severely degraded – why?
 - ENQDMN throughput hampered
 - JRNDMN sync rates capped
- Switch to client-server using ECP problem alleviated – why?
 - Block trading moved to network layer (less than 1ms)
 - Write Daemon asynchronous to user processes
 - Enhancements to ECP to bundle/defer journal syncs

Example 2

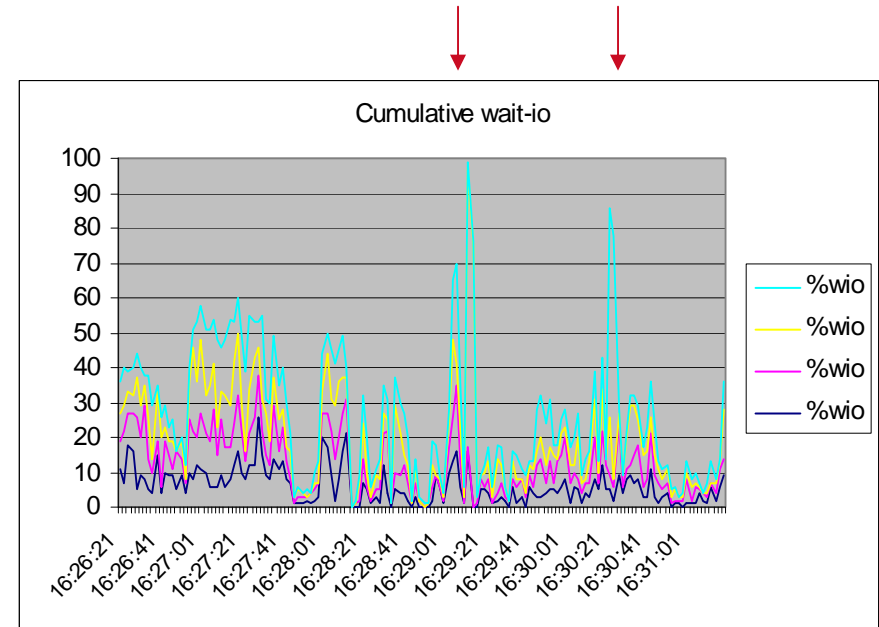
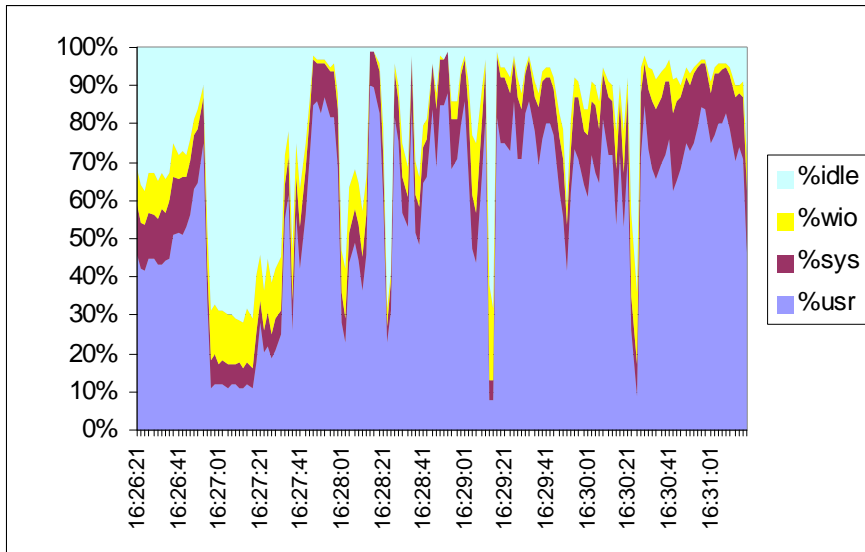


- Upgraded machine from dual single-core 2.8Ghz to dual quad-core 2.0Ghz
- Performance of nightly batch job *REDUCED* by 30%
- Disk utilization suffered during day because more jobs are “runnable” concurrently
 - Available buffers dropped to ZERO causing more IO (less cache efficiency)
 - Write daemon suspended database because it could not complete write cycle in 5 minutes

Example 3



Initial proof of concept too slow.



- Two wait-io spikes are WD runs.
- All files were on same disk (WIJ, Journals, Databases)
- Don't just check overall cpu data.
- Split the I/O.

Example 4

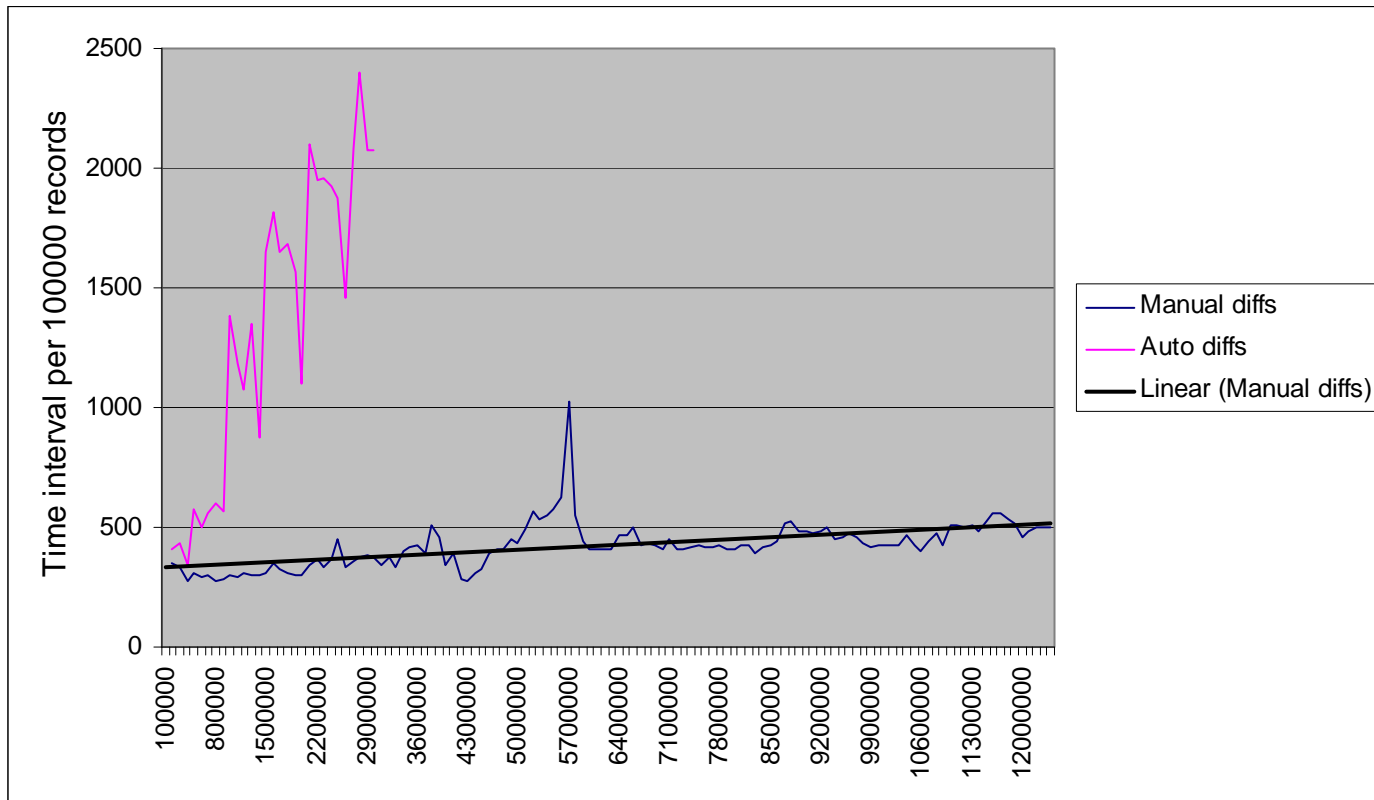


- Upgrade from 5.0.21 to 2007.1 slowed re-index by 4x
 - Windows 32bit to Windows 64bit upgrade
 - 4x dual core Xeon, 32GB memory, RAID 5 storage
 - WHY?
- What is a re-index operation actually doing?
 - Full table scan
 - Building a new random structure for the index
 - As the scan progresses more of the index needs to stay in memory
 - Single job
- Also, while 5.0.21 was tuned, 2007.1 was left on auto, so fewer buffers were allocated.

Example 4 (continued)



- A re-index is basically an I/O test, so the more memory you have available the less I/O you need to do.



Performance Tuning



Its an art and a science.

Please Use WRC Performance Team

Thank you. Merci. Dank U.