# Hyper-Threading Performance with Intel CPUs for Linux SAP Deployment on ProLiant Servers

## Session #3798

Hein van den Heuvel
Performance Engineer
Hewlett-Packard

# Topics

- Hyper-Threading Intro
- Implementation details Intel, IBM, Sun
- Linux implementation
- My own tests
- SAP (SD) benchmark
- Benchmark Results
- Conclusions: (18% improvement for SAP 2-tier)

# Intel Hyper-Threading Overview

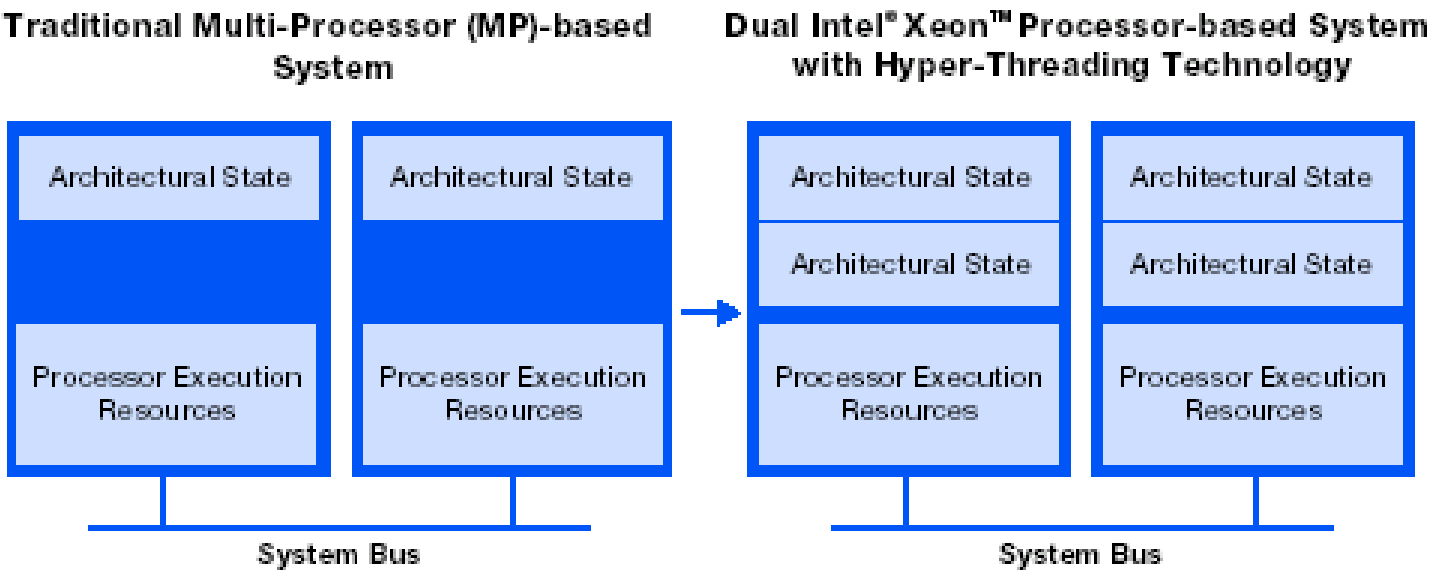
"Hyper-Threading Technology is a form of simultaneous multithreading technology (SMT), where multiple threads of software applications can be run simultaneously on one processor.

This is achieved by duplicating the architectural state on each processor, while sharing one set of processor execution resources. The architectural state tracks the flow of a program or thread, and the execution resources are the units on the processor that do the work: add, multiply, load, etc. "

http://www.intel.com/business/bss/products/hyperthreading/server/ht_server.pdf

http://www.intel.com/technology/hyperthread/

# Intel HT in a picture



To-be-updated

# Hyper-Threading Versus Dual Core

- HP (PA + ipf) opted for ‘dual core’ technology.
  - Each processor has full set of resources
  - Only limitation is shared ‘system’ connection.
  - Allows for dense (8p – 4u – 4640)
  - minimally constrained systems
- Software licensing impact (Oracle!)
- Hyper-Threading technology effectiveness will depend on application

# IBM P5 SMT Summary

**Enhanced Simultaneous Multi-Threading features**

To improve SMT performance for various workload mixes and provide robust quality of

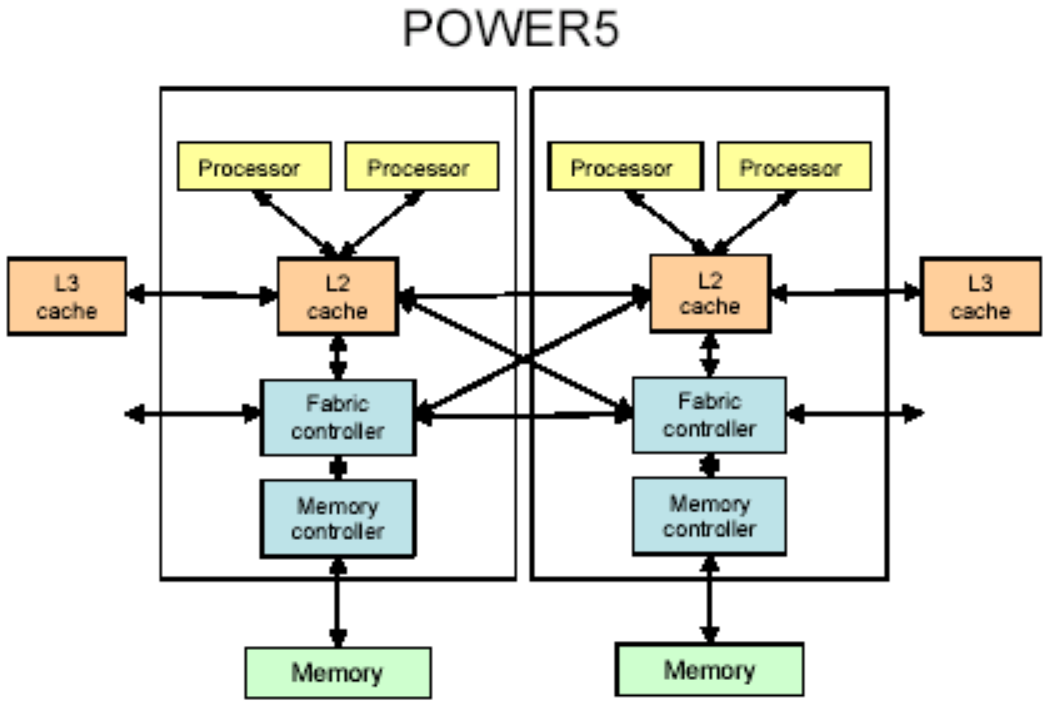
service, POWER5 provides two features:

- Dynamic resource balancing
  - – The objective of dynamic resource balancing is to ensure that the two threads executing on the same processor flow smoothly through the system.
  - – Depending on the situation, the POWER5 processor resource balancing logic has a different thread throttling mechanism.
- Adjustable thread priority
  - – Adjustable thread priority lets software determine when one thread should have a greater (or lesser) share of execution resources.
  - – The POWER5 supports eight software-controlled priority levels for each thread.

( http://www.redbooks.ibm.com/redpapers/pdfs/redp9117.pdf )

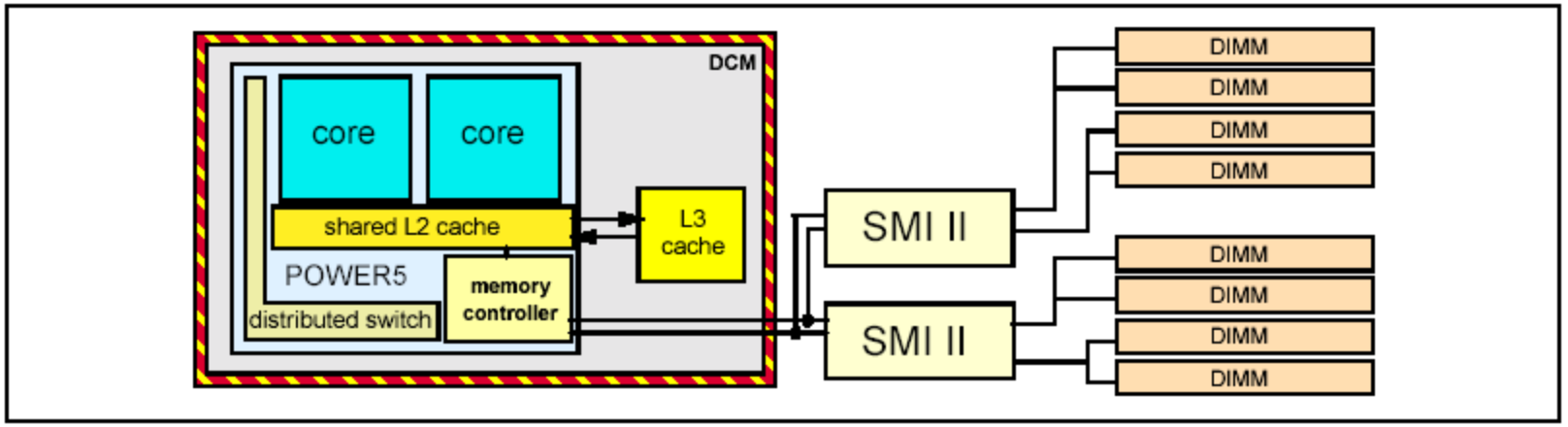
( http://www-1.ibm.com/servers/eserver/iseries/perfmgmt/pdf/SMT.pdf )

# IBM P5 Picture




A single die contains two identical processor cores, each supporting two logical threads. This architecture makes the chip appear as a four-way symmetric multiprocessor to the operating system. The POWER5 processor core has been designed to support both enhanced simultaneous multi-threading (SMT) and single-threaded (ST) operation modes.

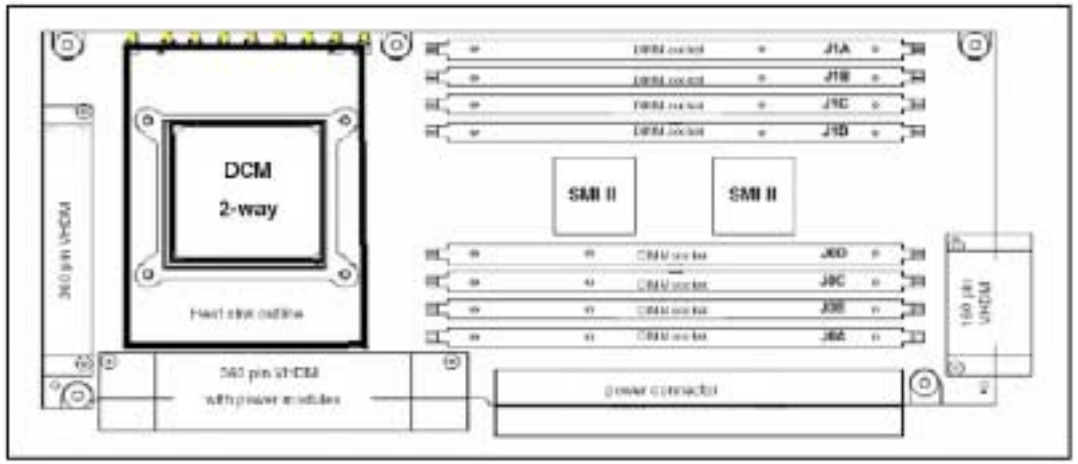
# IBM P5 Picture

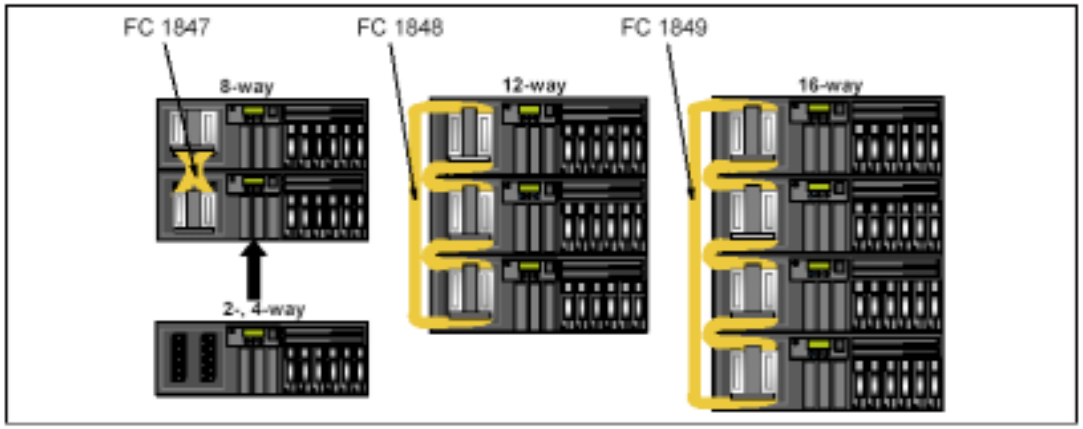



In the p5-570 system, the POWER5 chip has been packaged with the L3 cache chip into a cost-effective Dual Chip Module (DCM) package. The storage structure for the POWER5 processor chip is a distributed memory architecture that provides high-memory bandwidth. Each processor can address all memory and sees a single shared memory resource. As such, a single DCM and its associated L3 cache and memory are packaged on a single processor card.

Access to memory behind another processor is accomplished through the fabric buses. The p5-570 supports up to two processor cards (each card is a 2-way) in any building block. Each processor card has a single DCM containing a POWER5 processor chip and a 36 MB L3 module.

# IBM P5 Picture



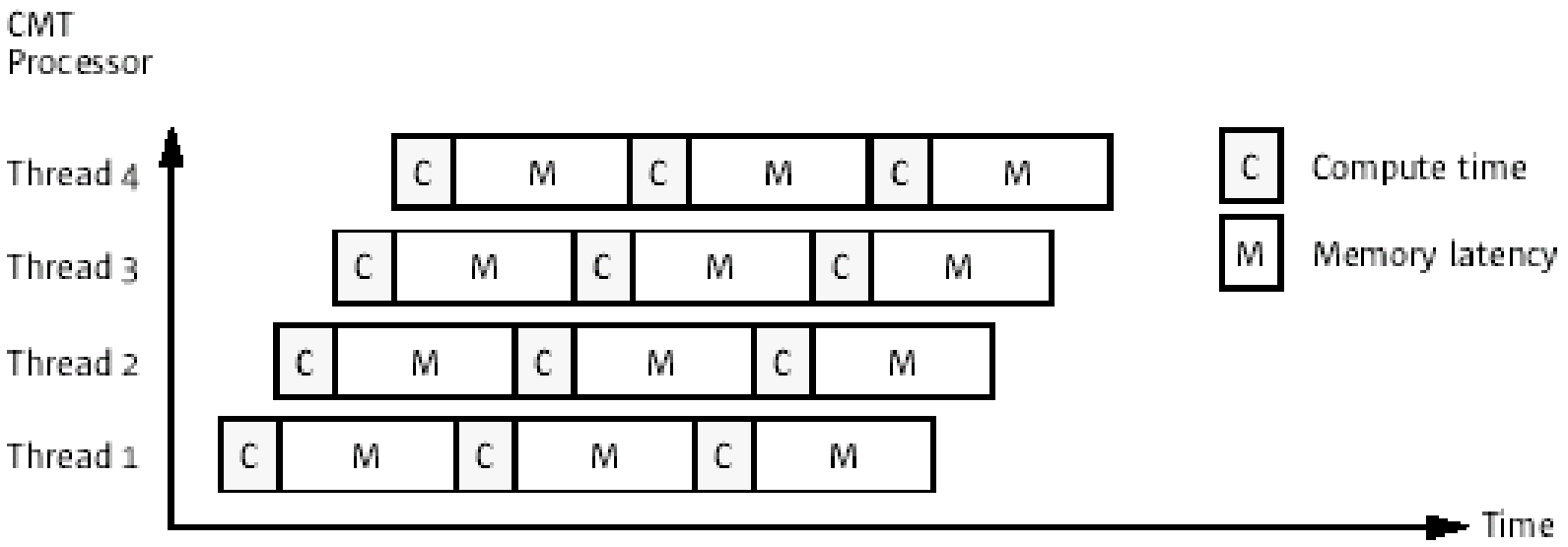
# SUN Summary

"Starting with the ability to run two concurrent threads in the UltraSPARC IV processor, support for multi-threading at the chip level will eventually enable single processors to process tens of threads simultaneously."

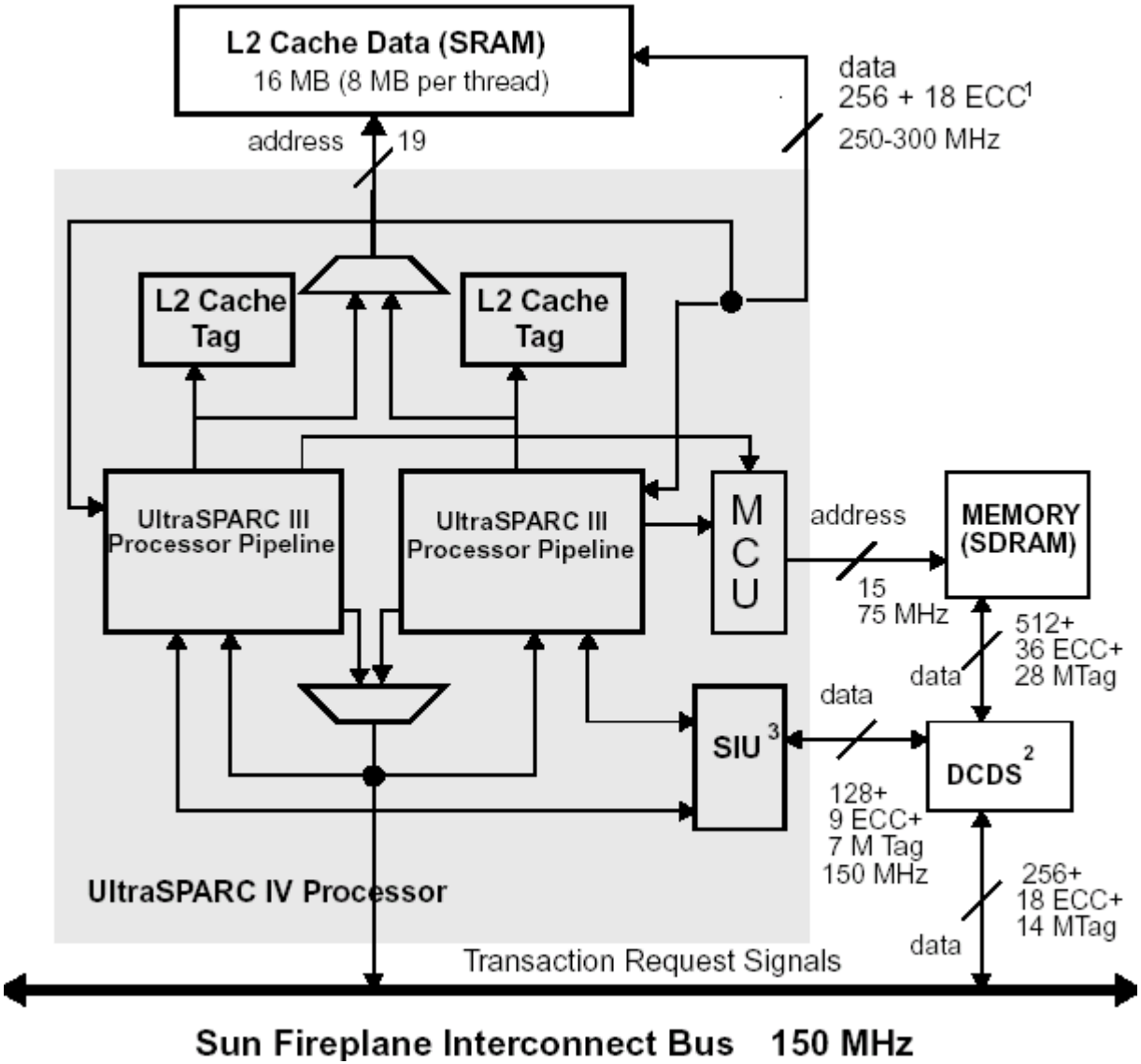
" Each Sun Fire Enterprise server supports multiple chip multi-threaded UltraSPARC IV processors with each processor capable of running up to two concurrent threads, providing up to twice the throughput of previous-generation UltraSPARC® III processors. Up to 72 UltraSPARC IV processors are supported in the high-end Sun Fire E25K server for support of up to 144
concurrent threads."

http://www.sun.com/servers/highend/whitepapers/ - Sun_Fire_Enterprise_Servers_Performance.pdf

# SUN Scheduling Picture



# SUN Picture


L2 Cache Data (SRAM)
16 MB (8 MB per thread)
address
19
data
256 + 18 ECC[1]
250-300 MHz
L2 Cache Tag
L2 Cache Tag
UltraSPARC III Processor Pipeline
UltraSPARC III Processor Pipeline
M C U
address
15
75 MHz
MEMORY (SDRAM)
data
512+
36 ECC+
28 MTag
SIU [3]
data
DCDS [2]
128+
9 ECC+
7 M Tag
150 MHz
256+
18 ECC+
14 MTag
data
UltraSPARC IV Processor
Transaction Request Signals
Sun Fireplane Interconnect Bus 150 MHz

# Linux Scheduler

- A Hyper-threaded CPU is seen by the kernel as two different CPUs so Linux does not have to be explicitly made aware of it.
  - Eprom config
- Linux breaks the “oldest idle rule” and forces immediate rescheduling when it discovers that a hyper-threaded CPU is running two idle processes
  - Oldest idle = Among the idle processors that can execute a given runnable process, the scheduler selects the least recently active”

# LINUX /proc/cpuinfo example

SuSE SLES 8(UnitedLinux 1.0)(i586) Kernel 2.4.21-138-smp (0).

```
# cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 15
model           : 2
model name      : Intel(R) Xeon(TM) MP CPU 3.00GHz
stepping        : 6
cpu MHz         : 2990.372
cache size      : 512 KB
physical id     : 0
:
processor       : 1
physical id     : 0
:
processor       : 2
physical id     : 1

processor       : 3
physical id     : 1
:
:
processor       : 4
physical id     : 2
:
processor       : 5
physical id     : 2
:
processor       : 6
physical id     : 3
:
processor       : 7
physical id     : 3
```

# LINUX /proc/cpuinfo example 2.4.19

```
SuSE SLES 8 (UnitedLinux 1.0) (i586)\nKernel 2.4.19-64GB-SMP (2)"

cat /proc/cpuinfo
:
flags     : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm


and NO HT shows:


processor   : 0
physical id : 0
processor   : 1
physical id : 0
processor   : 2
physical id : 0
processor   : 3
physical id : 0
```

# LINUX cpu busy indication

- CPU time no longer behaves linearly.
- Example for 4 CPU HT system: 8 virtual processors.
  - Start executing 4 tasks each 100% cpu busy.
  - Scheduler will push those to run on 4 physical processors
  - TOP and VMSTAT will show 50% CPU busy (1/2 processors!)
  - But adding 4 more tasks will NOT give 2x throughput

# What would benefit from Xeon HT?

- Yes:
  - Multi Threaded apps.
  - Concurrent single threaded
  - Need some (low level) STALLs that would otherwise have CPU essentially unused.
    - BTW… a CPU waiting for memory to come in while really 'idle' for the application is 100% CPU busy for the OS.
- No:
  - Single threaded application
  - CPU intense, L1/L2 Cache effective apps.

# Linux processor binding tools

•int **sched_setaffinity** ( pid_t pid, unsigned int len,
unsigned long *new_mask_ptr)

•int **sched_getaffinity** ( pid_t pid, unsigned int *user_len_ptr,
unsigned long *user_mask_ptr)

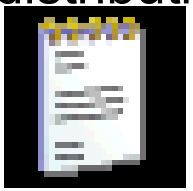
• Tool by Robert Love: affinity-run.c

• http://www.kernel.org/pub/linux/kernel/people/rml/cpu-affinity/affinity-run.c

• My variation on that tool with get/setpriority included

•.H file in case your distribution has improperly exported syscalls.

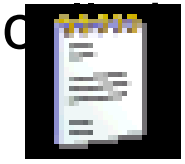
C File



H File

# My own tests

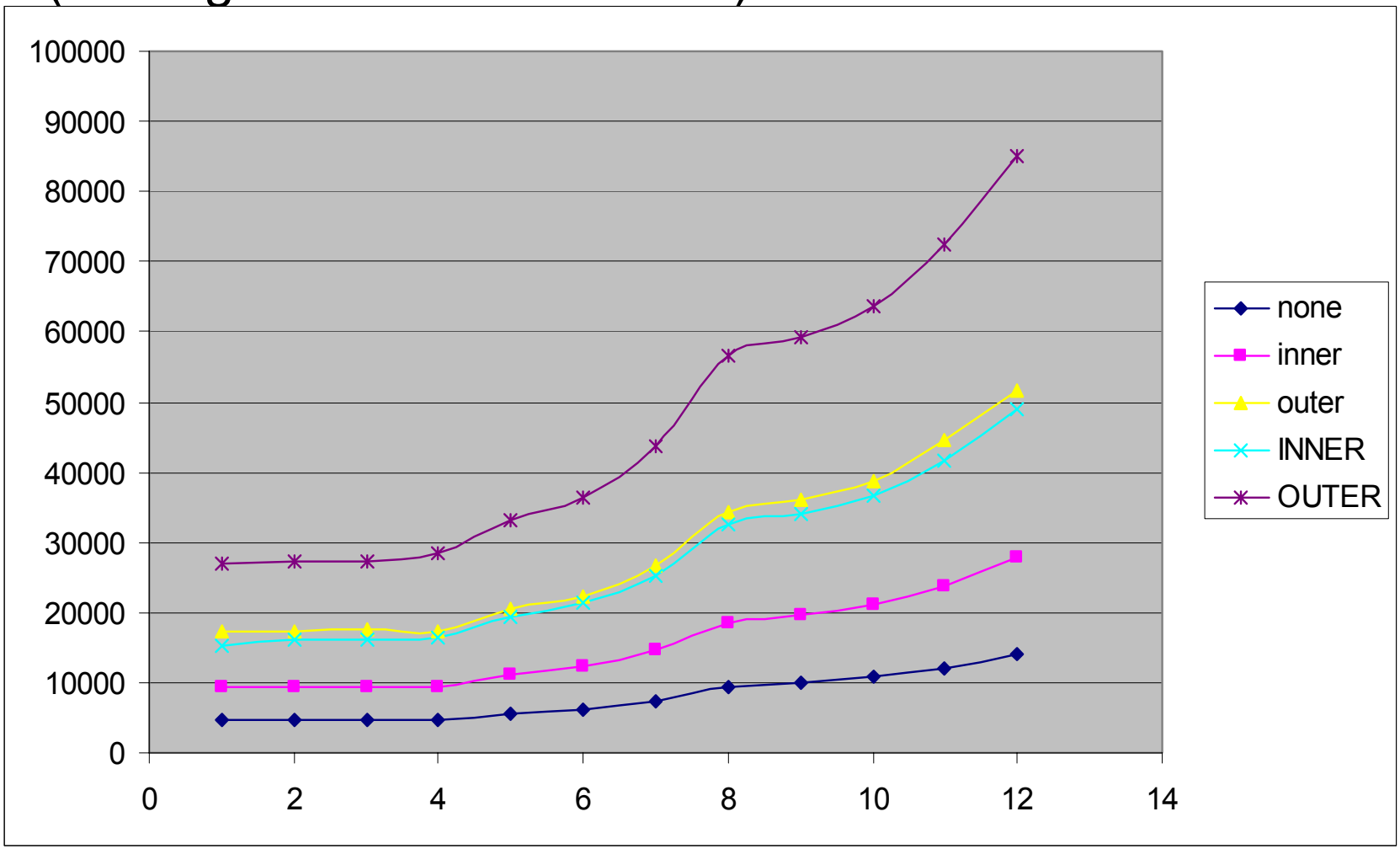
- Process 10 loops over 100 MB  Array
  - 10,000 x 10,000 so ‘outer’ jumps physical memory pages
- Measure run-time in milliseconds, Lower is Better
- 5 sub-test
  - Inner  loop first, read-only.
  - Outer  loop first, read-only.
  - INNER  loop first, read + write
  - OUTER  loop first, read + write
  - No c[illegible]ange in loop, read-only



C File

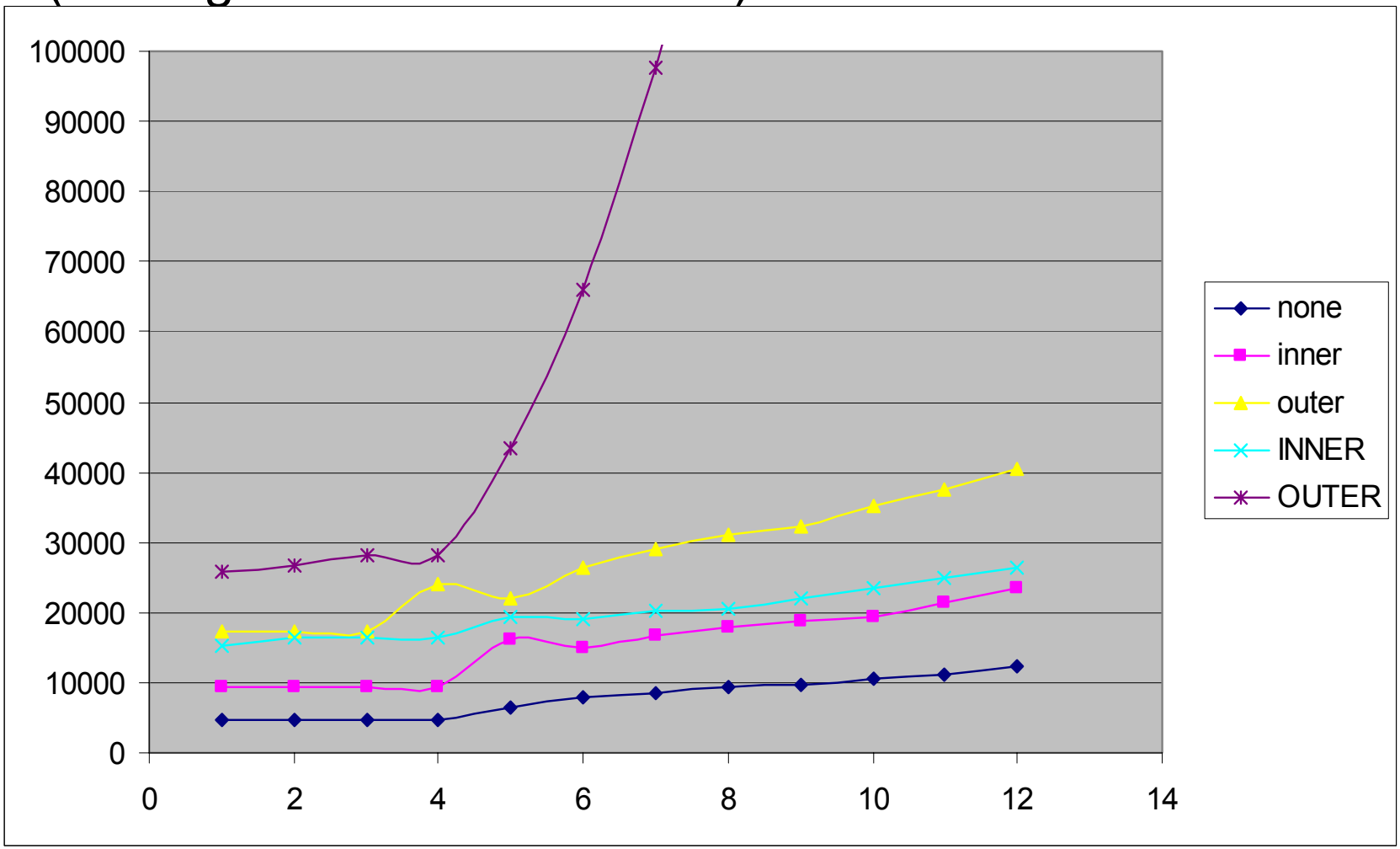
# 1-12 streams on 4 CPUs NO HT

(Average time. Less is better)



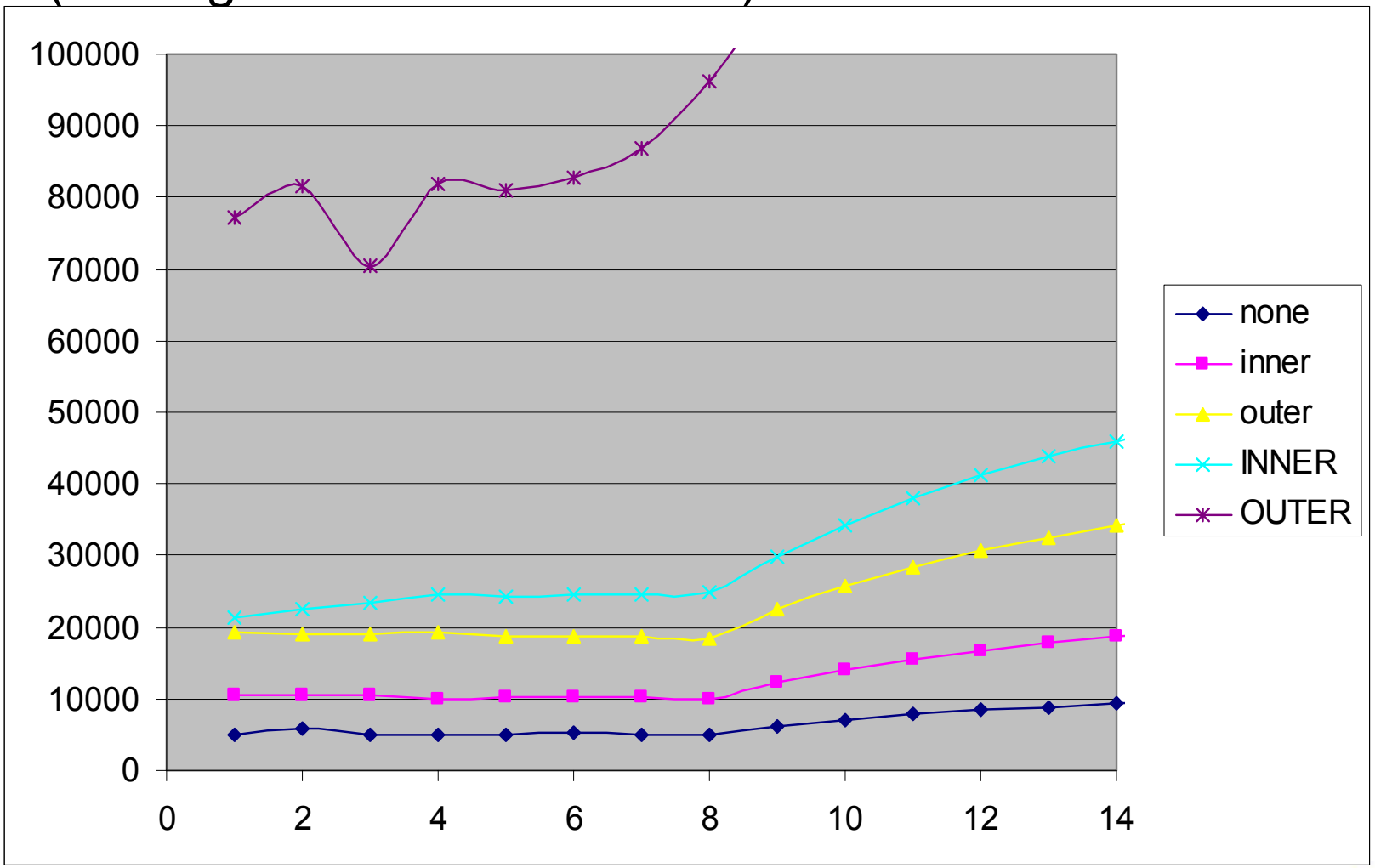
# 1-12 streams on 4 CPUs HT

(Average time. Less is better)



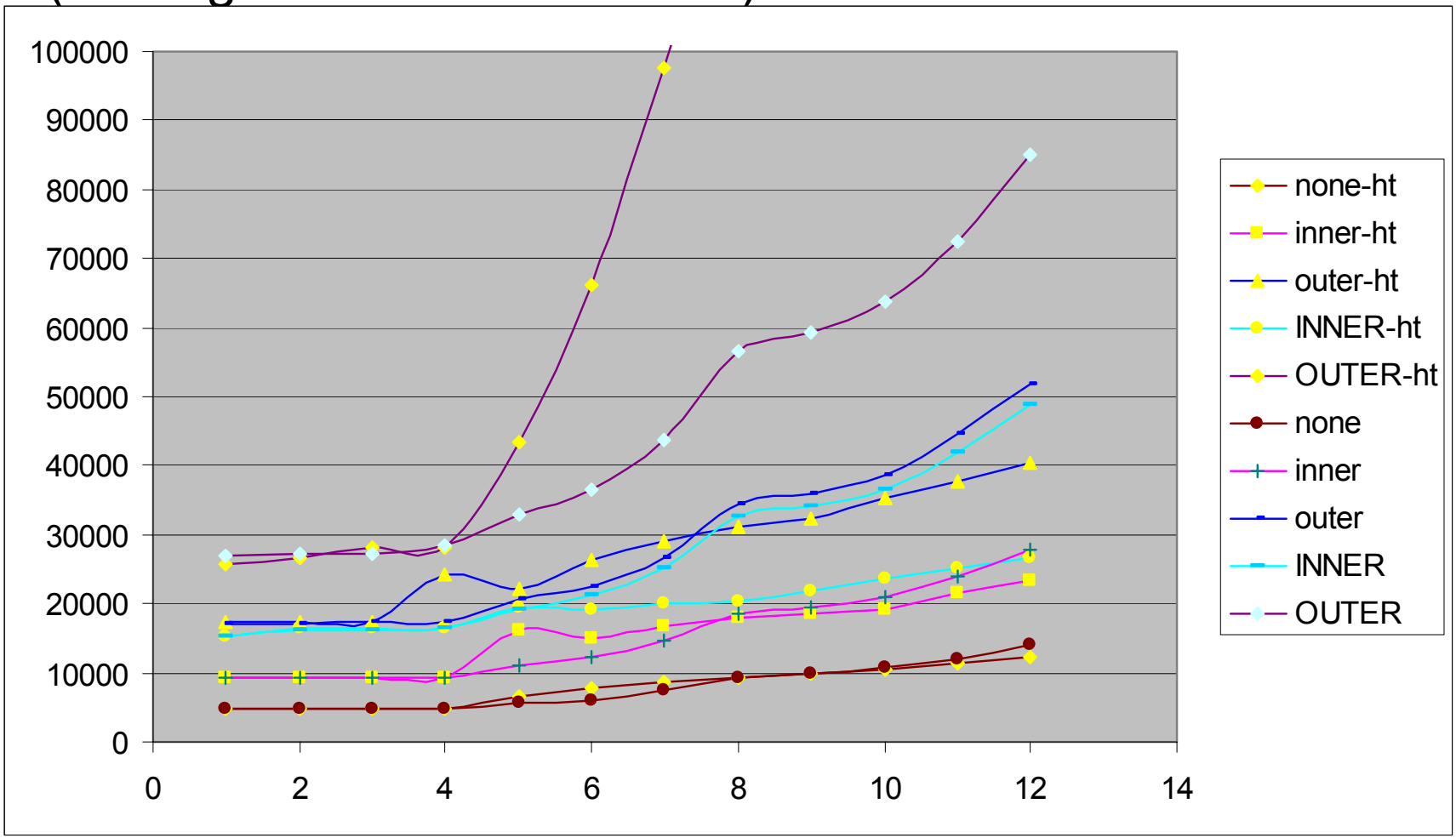
# DL760 with 8 real CPUs

(Average time. Less is better)



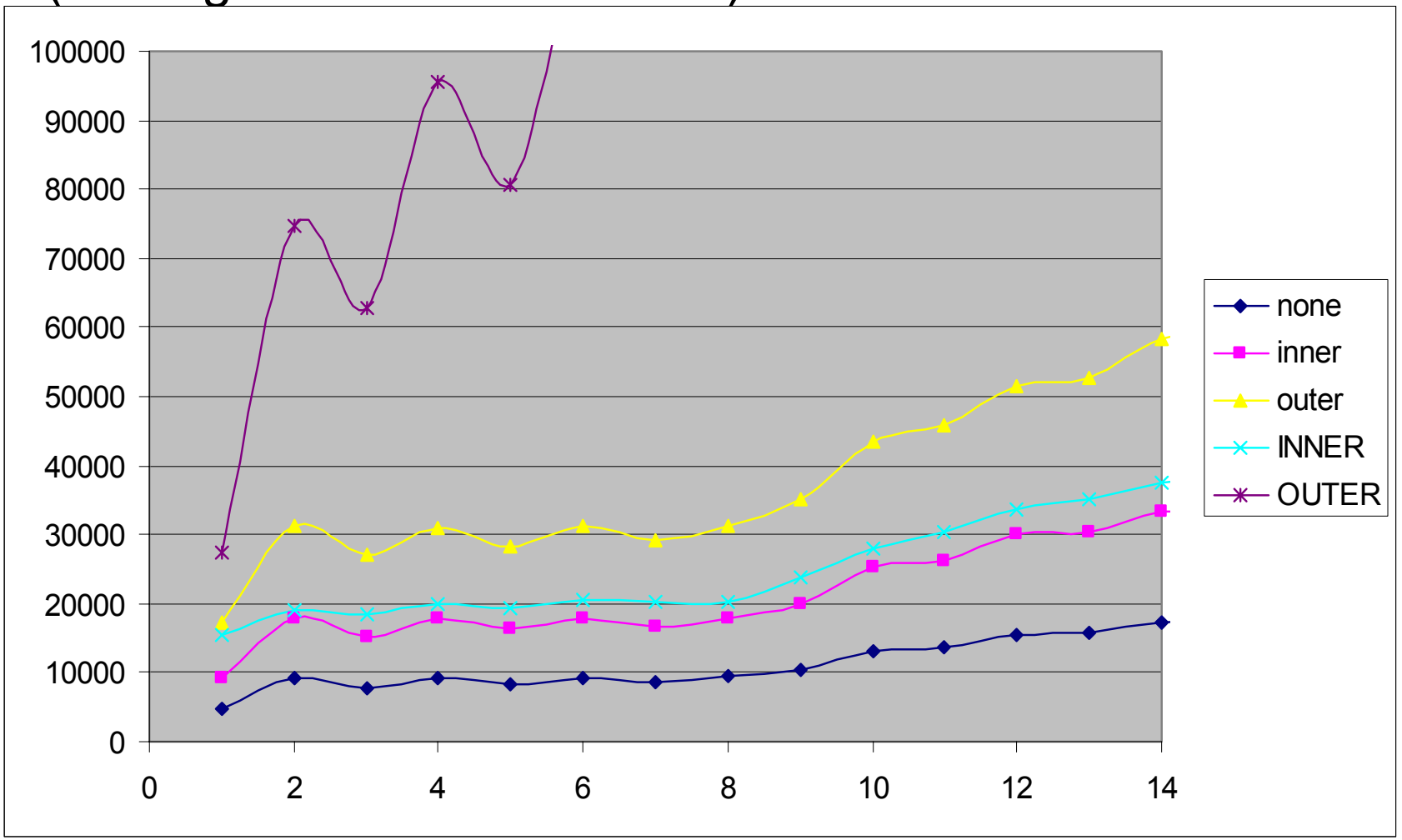
# 1-12 streams on 4 CPUs (mixed)

(Average time. Less is better)

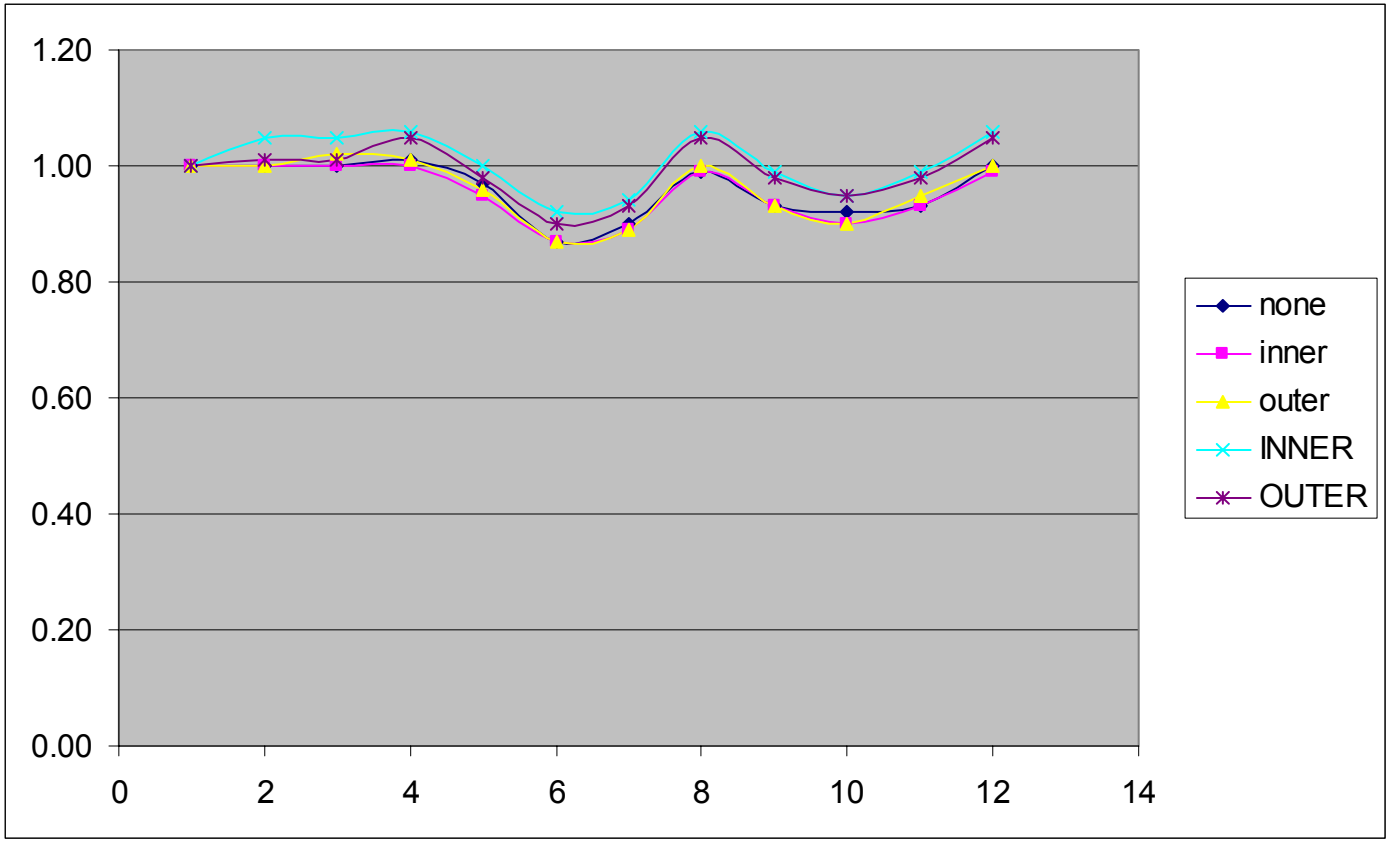


# Hyper-Threading and BAD Affinity

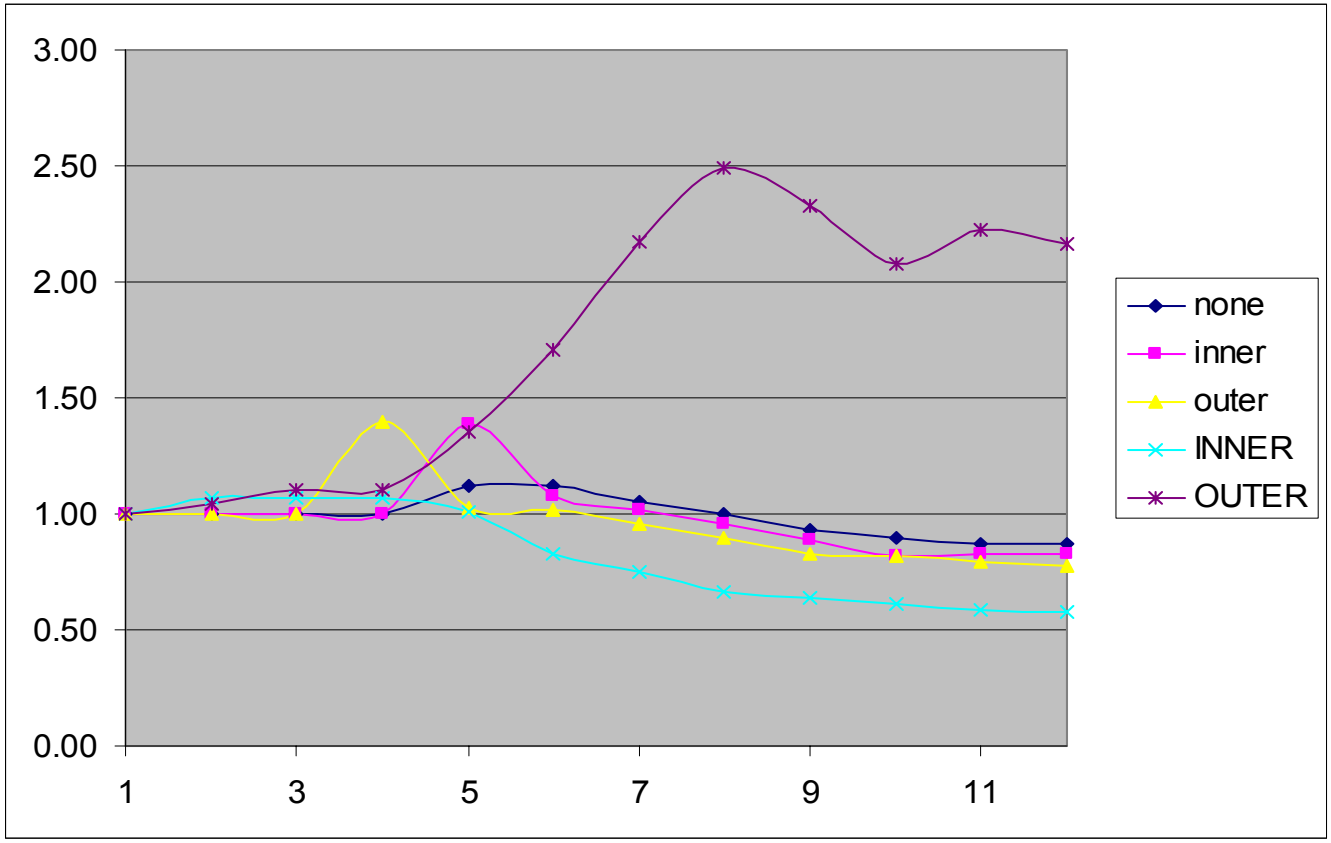
(Average time. Less is better)



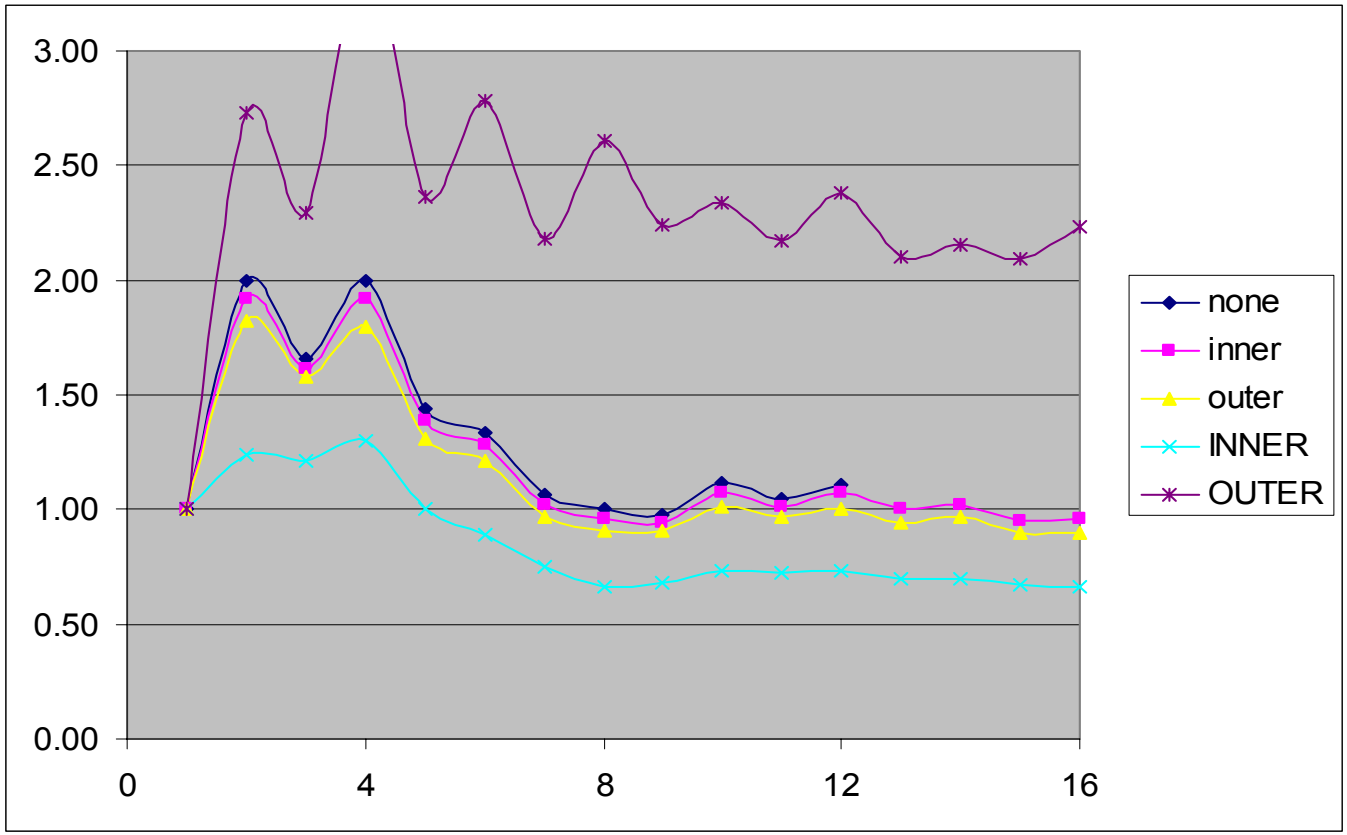
# 1-12 streams on 4 CPUs (relative)



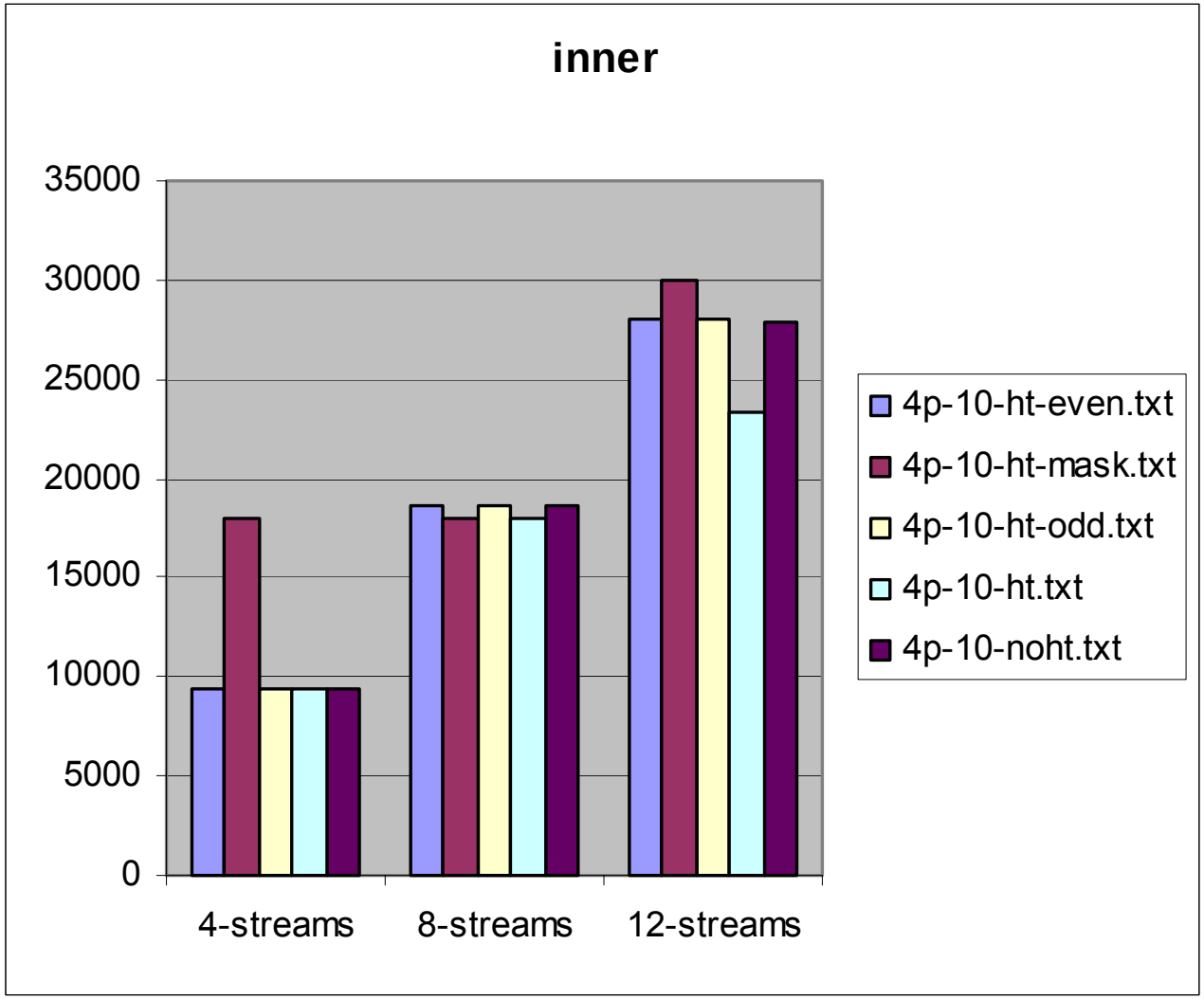
# 1-12 streams on 4 CPUs HT (relative)



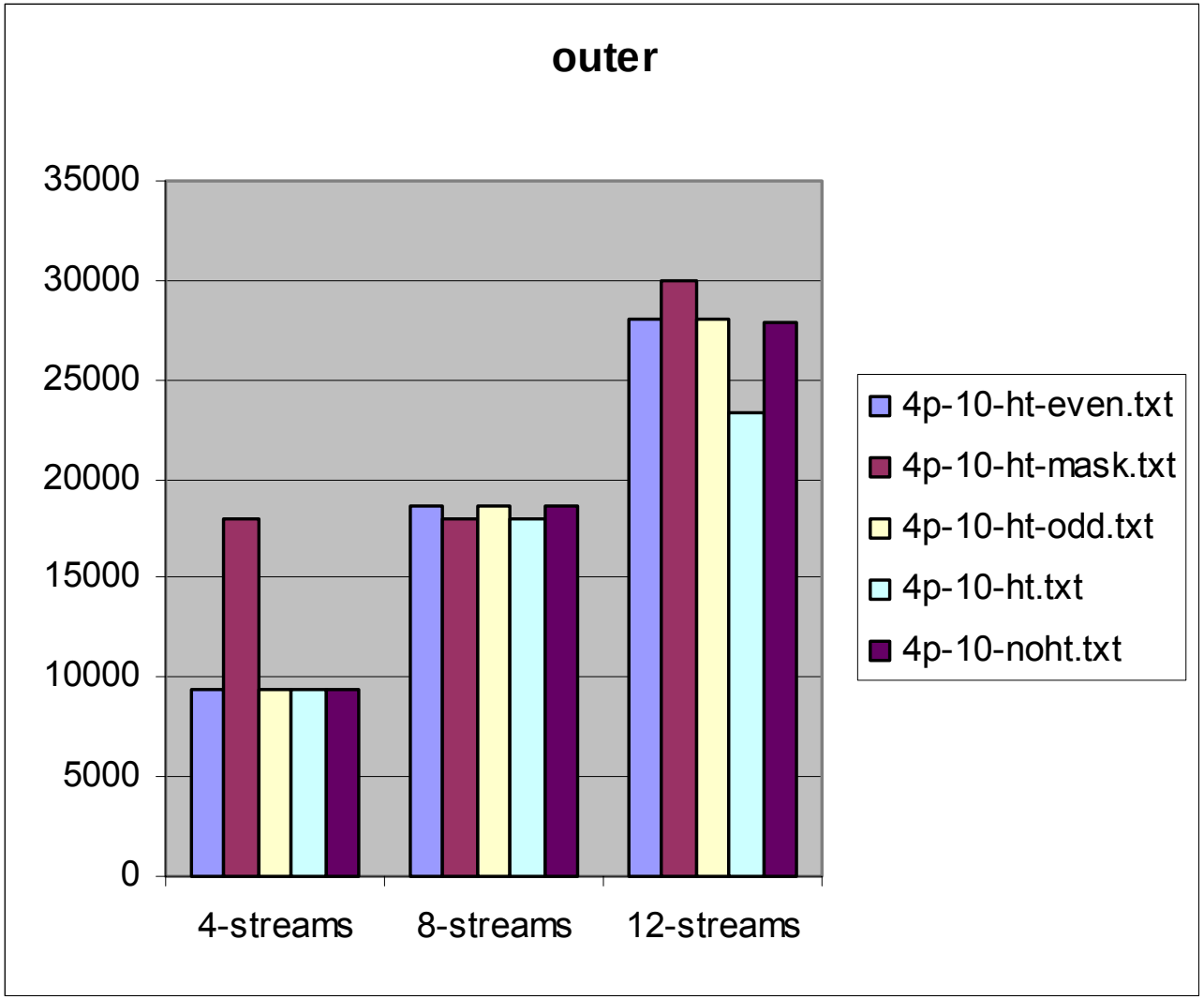
# Hyper-Threading and BAD affinity (relative)



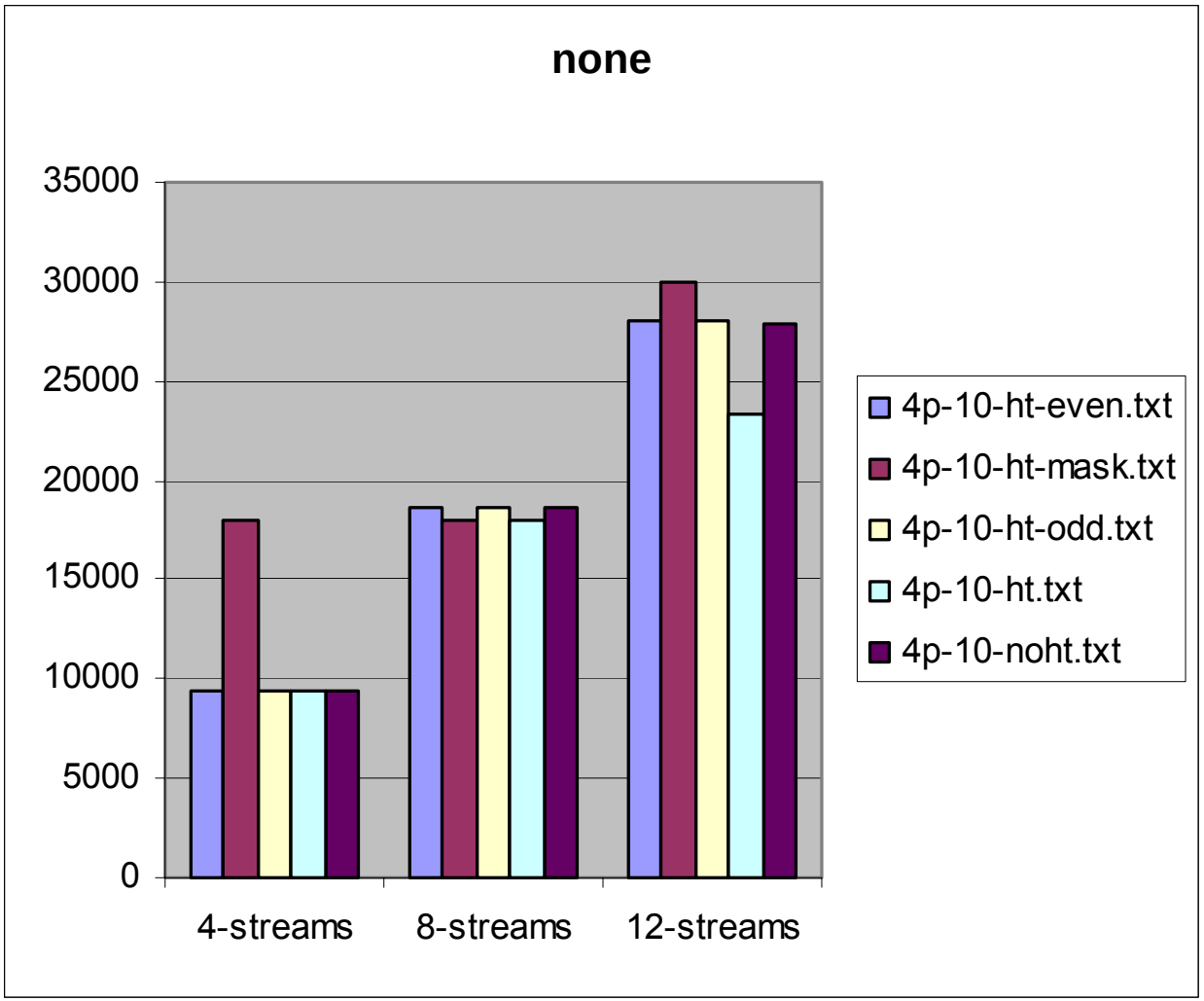
# Inner-loop first test



# Outer-loop first test



# No array access test



# Raw data from tests

| Procs | none-ht | none | | inner-ht | inner | | outer-ht | outer | | INNER-ht | INNER | | OUTER-ht | OUTER |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 4685 | 4684 | | 9357 | 9363 | | 17241 | 17169 | | 15365 | 15387 | | 25719 | 26922 |
| 2 | 4686 | 4684 | | 9362 | 9365 | | 17225 | 17163 | | 16386 | 16172 | | 26647 | 27303 |
| 3 | 4685 | 4684 | | 9358 | 9363 | | 17243 | 17467 | | 16405 | 16179 | | 28268 | 27263 |
| 4 | 4690 | 4709 | | 9362 | 9398 | | 24127 | 17303 | 40 | 16498 | 16376 | | 28248 | 28386 |
| 5 | 6561 | 5697 | 16 | 16234 | 11070 | 47 | 22123 | 20600 | 8 | 19370 | 19271 | | 43312 | 32998 |
| 6 | 7862 | 6132 | 29 | 15093 | 12188 | 24 | 26422 | 22396 | 18 | 19031 | 21293 | 10 | 66050 | 36426 |
| 7 | 8647 | 7397 | 17 | 16741 | 14568 | 15 | 28906 | 26626 | 9 | 20199 | 25256 | 20 | 97666 | 43635 |
| 8 | 9336 | 9312 | | 17975 | 18587 | 3 | 30993 | 34305 | 9 | 20389 | 32615 | 37 | 128282 | 56479 |
| 9 | 9801 | 9833 | | 18689 | 19571 | 4 | 32239 | 35939 | 10 | 22002 | 34129 | 35 | 134646 | 59248 |
| 10 | 10561 | 10716 | | 19284 | 20968 | 8 | 35280 | 38625 | 8 | 23587 | 36634 | 35 | 133578 | 63651 |
| 11 | 11252 | 11918 | 5 | 21481 | 23853 | 9 | 37648 | 44695 | 15 | 25051 | 41777 | 40 | 157133 | 72568 |
| 12 | 12186 | 14025 | 13 | 23383 | 27890 | 16 | 40542 | 51671 | 21 | 26521 | 48869 | 45 | 166529 | 84976 |
| | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | |

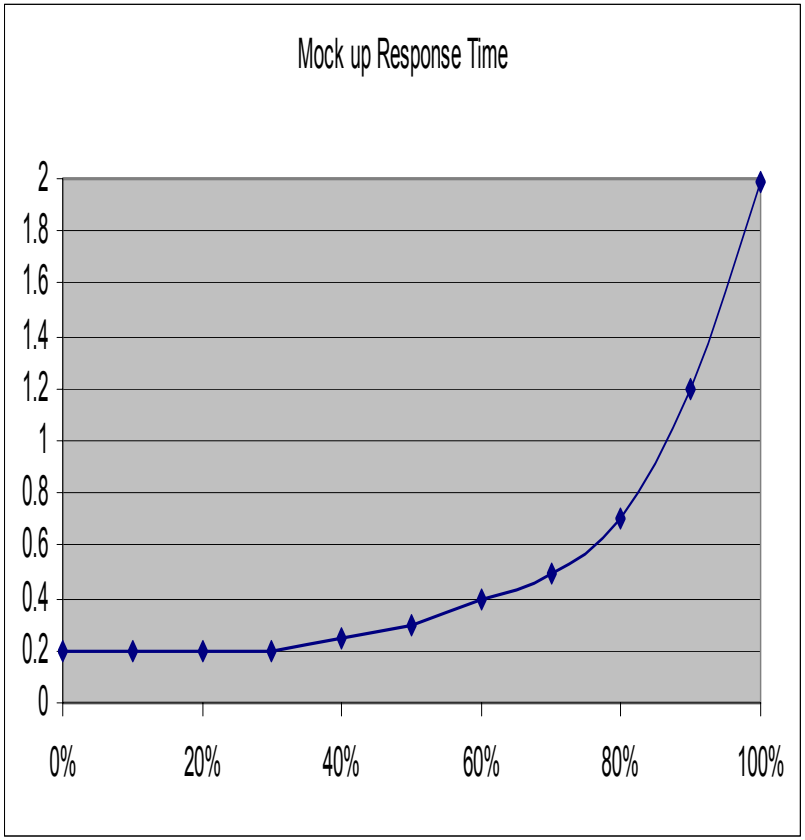
# SAP Architecture overview

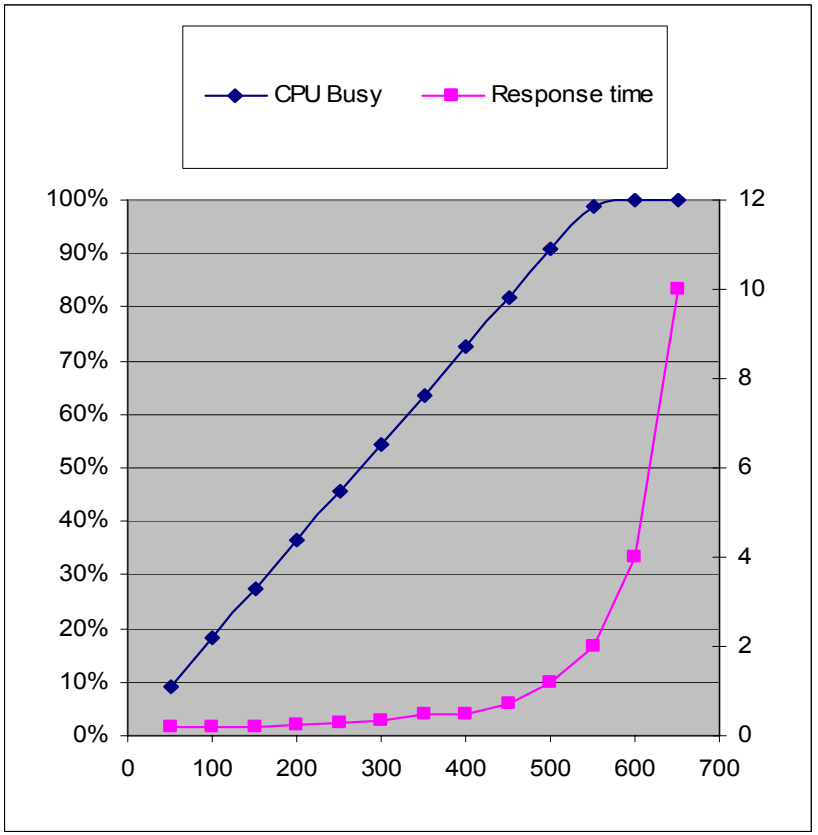
- Each application ‘Instances’ maintains own task(dispatch) queue:
  - DIAlogue, UPDate, ENQue, SPOol,…
- Multiple Instances is just fine, even on single box
  - Less contention on local resources (buffers)
- Dispatch Queue goes to first free worker ( “disp+work” ) starting at first worker.
- First worker only idle waiting for database (Oracle) or if there simply is no more new work.
- Used ‘affinity’ to force that critical processes did not share same physical processor. (Update!)

# SAP SD Benchmark Overview

- Official measurements is ‘SAPs’ which corresponds to a certain throughput (Dataprocessing Steps / Hour)
- Practical measurement is in Users
  - Each user executes a fixed (20) number of DS steps in a loop with 10 second think time and 2.0 second Response Time requirement.
- http://www.sap.com/benchmark/BM_description.htm#SD
- http://www.sap.com/benchmark/sd2tier.asp

# Response time Knee (Mock up)



# And now finally for the real work we did…

# SAP SD Result Matrix

| Hyper-Threading | Users | Resp Time | CPU time |
|---|---|---|---|
| On | 550 | 0.3 | 80% |
| On | 650 | 1.99 | 99% |
| Off | 550 | 1.99 | 99% |
| Off | 650 | X | 100% |

# System Under Test: DL560

- DL560
- 3.0 Ghz Processor
- 4GB Memory
- Sap 4.7 (620 kernel)
- Oracle as DB

# Benchmark Conclusions

- Hyperthreading gives 18% performance boost.
- Must takes extra steps to exploit fully
- Why not enable? Just do it.
- Not Dual Core.

# Resources

- Linux…
- HP…
- Intel…
- IBM…


invent

# HP WORLD 2004

Solutions and Technology Conference & Expo

Co-produced by:

interex

shared knowledge • shared power

encompass

AN HP USER GROUP

RECOMMENDED TRAINING VENUE FOR THE

HP Certified Professional