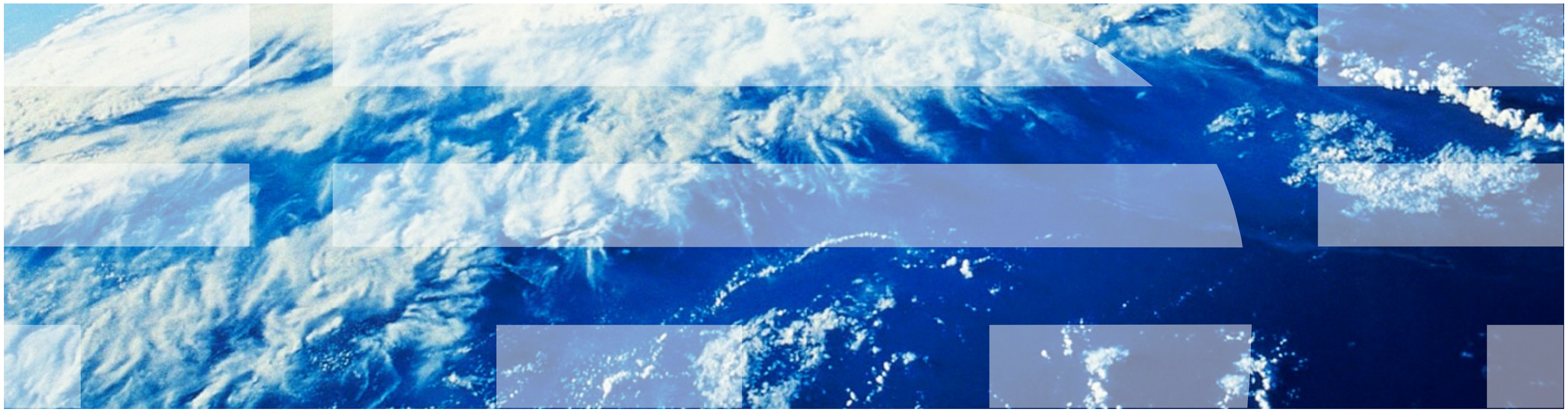
# HPC Tuning on POWER7



## Agenda

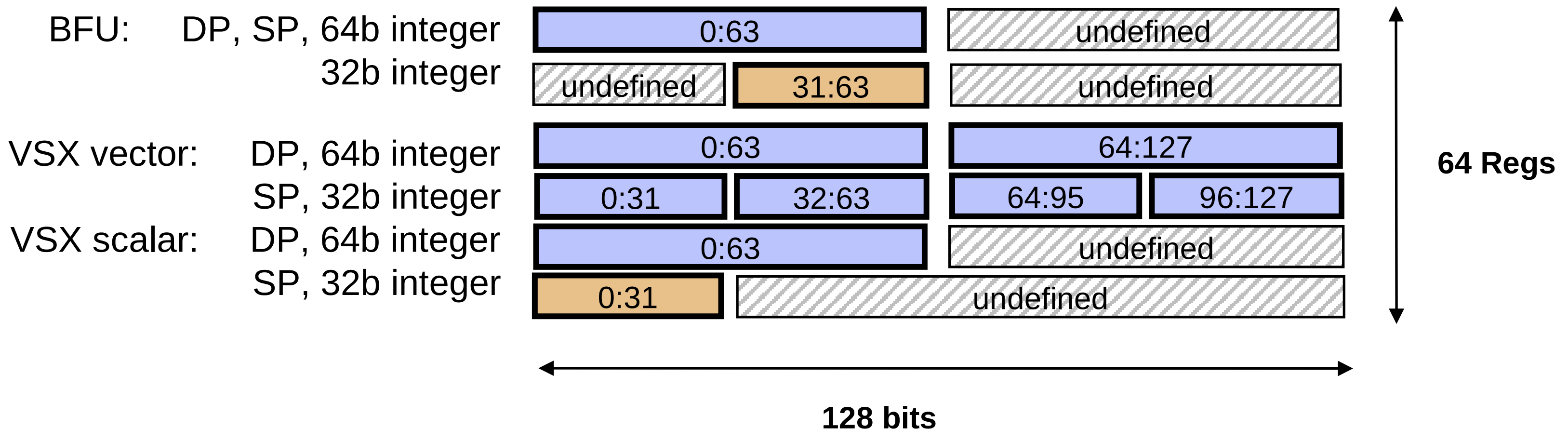
### Software

- Compiler optimization
- Vectorization
- Tuned libraries
- SMT
- Affinities
- Memory pages

### Applications

- STREAM_MPI
- MMULT
- RTM
- VMX

## VSX Registers



## One POWER7 Core
### SMT: Multi-threading Evolution

- Why ?
  - Latency to all levels of memory increasing
  - Algorithms and languages involve more cache misses

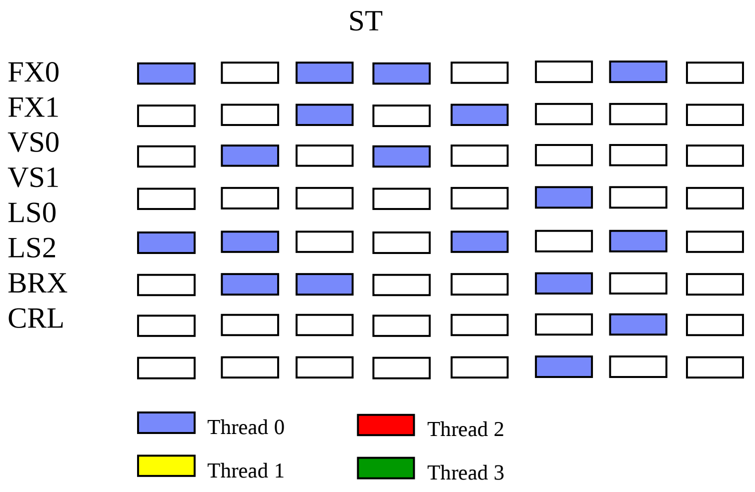
- Simultaneous Multi Threading (SMT) allows processor to exploit Instruction Level Parallelism (ILP) in the event of a stall
  - Stalls caused by :
    - DIVIDES, SQRT, branches, … on POWER5 and POWER7
    - Same story + High frequency & in order execution on POWER6
    - …

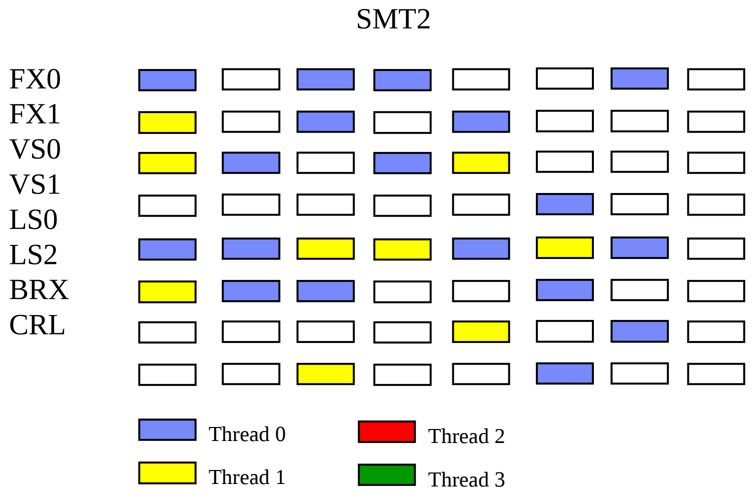
- SMT is supported on
  - POWER5 & POWER6 & POWER7

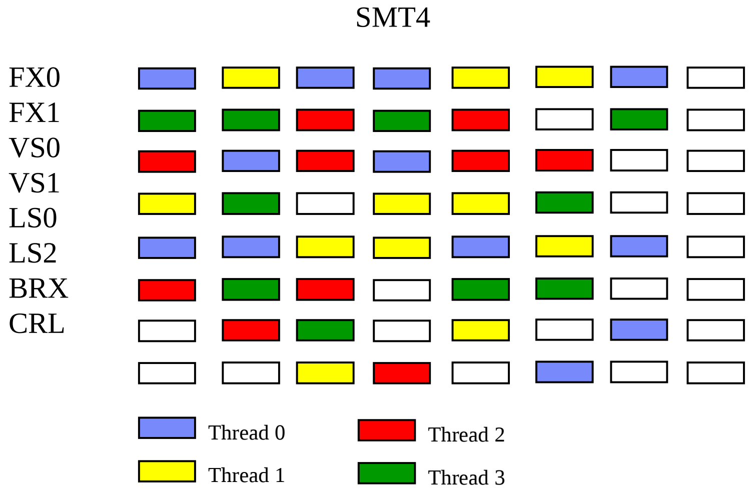
- Automatically activated by Operating System when needed
  - No changes to application
  - Transparent to users
  - Very easy to use

# One POWER7 Core

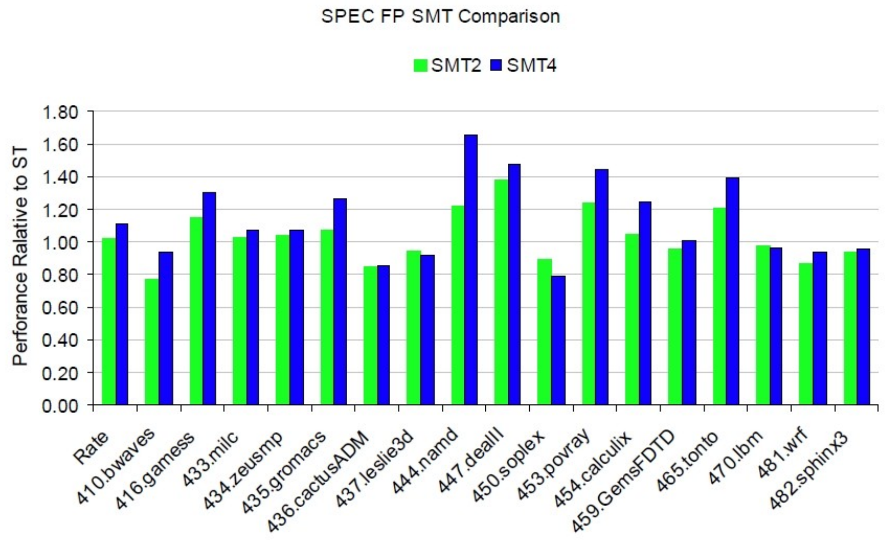
## SMT: Multi-threading Evolution

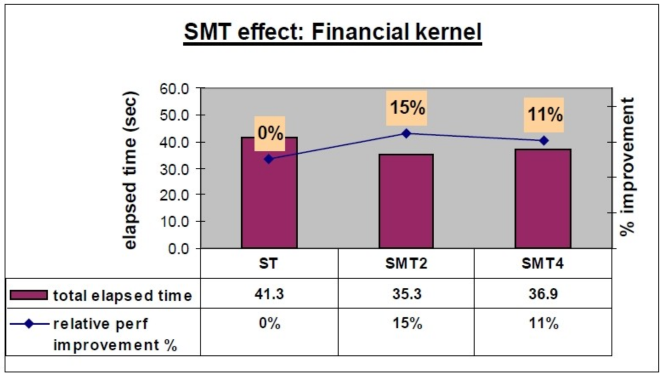




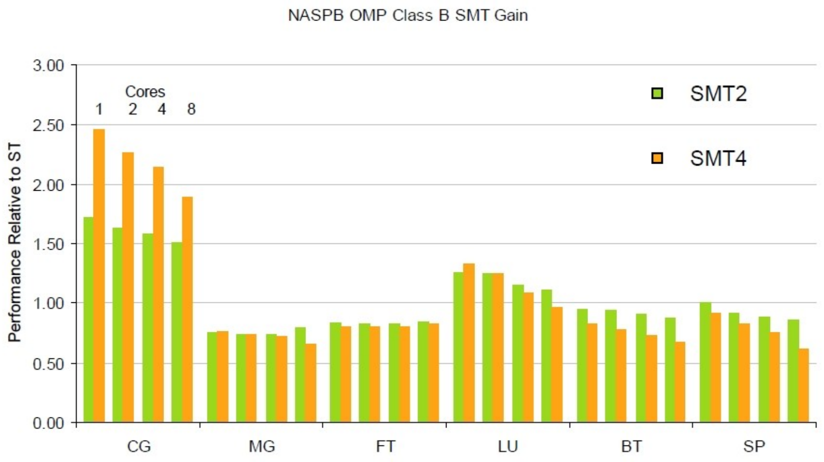


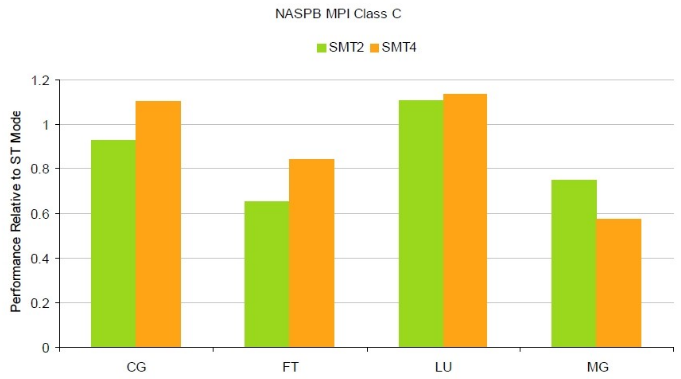
# One POWER7 Core
## SMT: Performance Examples





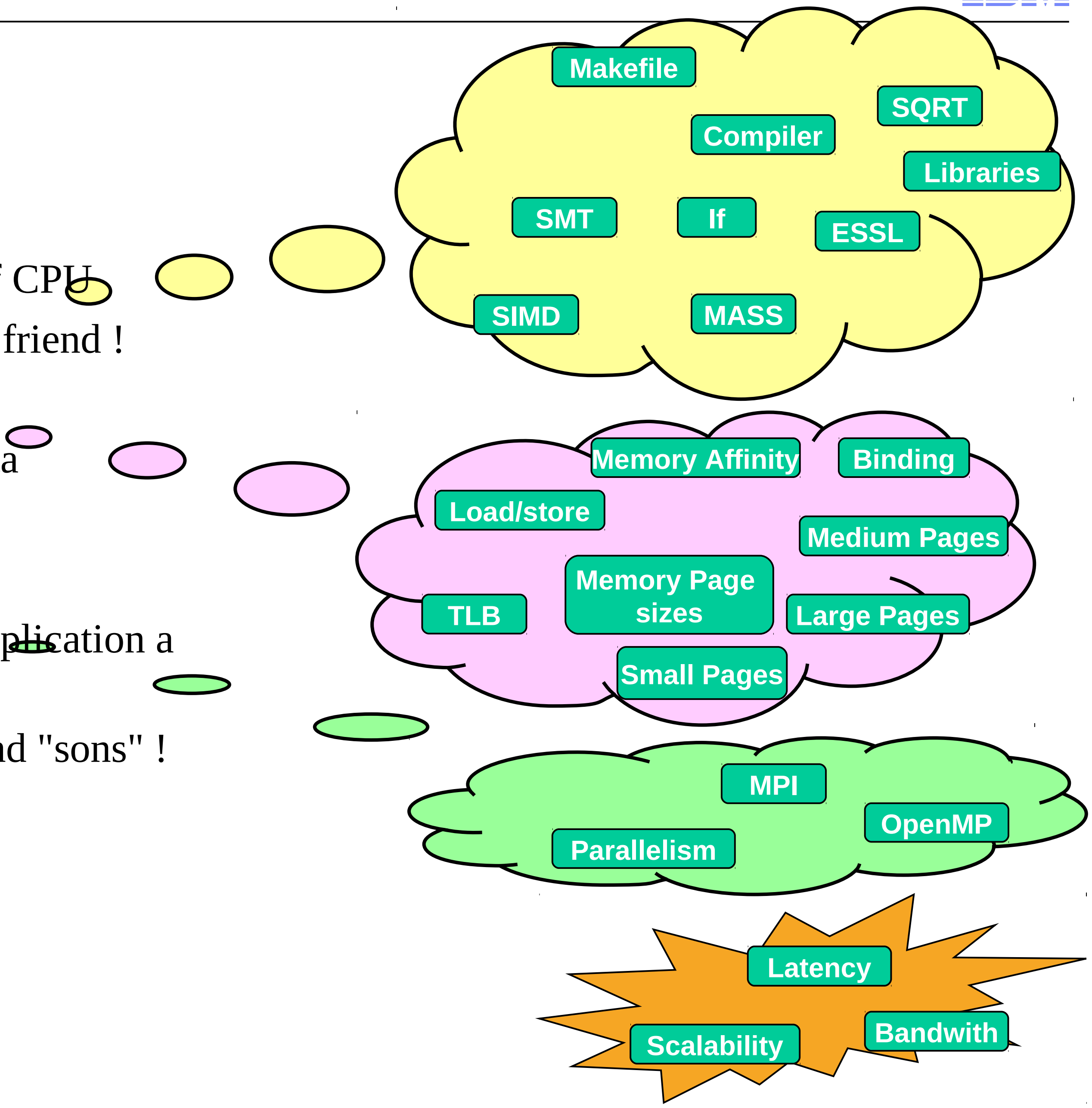
| | ST | SMT2 | SMT4 |
|---|---|---|---|
| total elapsed time | 41.3 | 35.3 | 36.9 |
| relative perf improvement % | 0% | 15% | 11% |





## Tuning Techniques

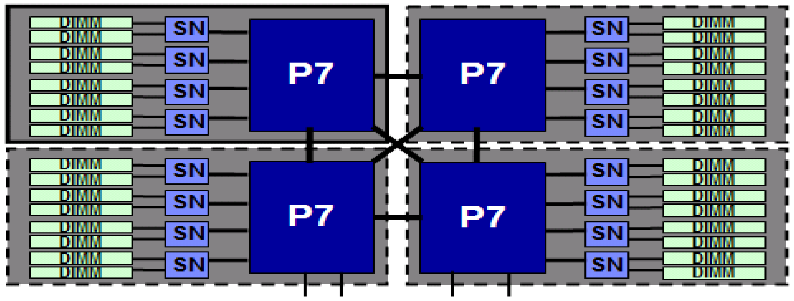
- Increase utilization of CPU
  - The CPU is your friend !
- Improve access to data
  - Feed the CPU !
- Offer your parallel application a good scalability
  - Use "brothers" and "sons" !



# Binding and Memory Affinity Motivations

Most HPC applications perform better if

– processes and threads don't change from one CPU to another during execution
  - **binding**
– data used is local
  - **memory affinity**



- On numa-like systems (where memory cards are connected to processors chips) the locality of data is very important from a performance point of view
  → use "binding" & "memory affinity"
  - IBM POWER5/6/7 based systems
  - AMD Opteron based systems

On real SMP systems it is less/not important: the path to memory location is the same for all CPUs.
→ "binding" & "memory affinity" not needed
- IBM BG/P
- Intel Woodcrest/Clovertown/Hapertown based systems.

## Binding and Memory Affinity Motivations

- The process/thread allocates memory on the physical processor card it's running on.
  - i.e. all memory used by the process/thread is local.

  → Improve the memory bandwith and latency

- Tools allow to manage memory rules
  - Commands: `numactl` on Linux, `vmo` on AIX
  - environment variables

- By default Operating Systems behavior is the same: **no binding**
  - OS scheduler decides to move or not processes/threads from one core to another based on its own policy

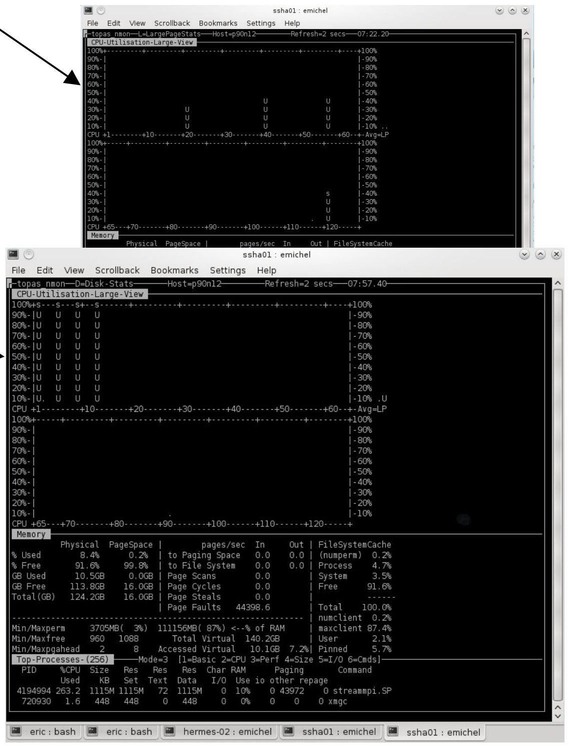
- Tools allow to manage binding behaviors
  - Commands: `taskset` on Linux, `execrset`, `bindprocessor` on AIX
  - environment variables

## One POWER7 Core: Tuning Techniques
## Affinities : How to bind processes

- AIX:
  Several ways to bind processes:
  - execrset command
    - execrset -c 0-7 -e <exe>
    - Will force the process <exe> to run on cores #0,1,2,3,4,5,6,7
  - Bindprocessor command and library
    - Bindprocessor PID procnum
  - Using POE runtime environment (for MPI jobs only)
    - Export MP_TASK_AFFINITY=core(2)
- Linux
  - taskset command is the best one (the only one ?)
    - taskset –c 0-2,5 <exe>
    - Will force the process <exe> to run on cores #0,1,2,5
- Both MPI and OpenMP binding techniques can be used for mixed mode programs

## One POWER7 Core: Tuning Techniques
## Affinities : How to bind OpenMP threads

- AIX & Linux:
  - OpenMP threads have to be bound using XLF environment variable:
    - export OMP_NUM_THREADS=4
      export XLSMPOPTS=startproc=0:stride=4
    - Will bind 4 OpenMP threads starting at core #0 with a stride of 4:
      threads bound on cores #0 4 8 12
  - New way comes with latest version of compiler
    - export XLSMPOPTS="procs=0,2,4,6"
  - Or choose binding over resources:
    MCM, L2CACHE, PROC_CORE, or PROC
    - export XLSMPOPTS="bind=PROC=0,16,2"
    - export XLSMPOPTS="bindlist=MCM=0,1,2,3"
    - export XLSMPOPTS="bindlist=L2CACHE=0,1,0,1,0,1,0,1"



```
-topas_nmon--L=LargePageStats---Host=p90n12---------Refresh=2 secs---07:22.20
 CPU-Utilisation-Large-View
100%+---------+---------+---------+---------+---------+---------+----+100%
90%-|                                                                |-90%
80%-|                                                                |-80%
70%-|                                                                |-70%
60%-|                                                                |-60%
50%-|                                                                |-50%
40%-|                                    U            U              |-40%
30%-|                U                   U            U              |-30%
20%-|                U                   U            U              |-20%
10%-|                U                   U            U              |-10% ..
CPU +1--------+10-------+20-------+30-------+40-------+50-------+60--+-Avg=LP
100%+-----+---------+---------+---------+---------+---------+--------+100%
90%-|                                                                |-90%
80%-|                                                                |-80%
70%-|                                                                |-70%
60%-|                                                                |-60%
50%-|                                                                |-50%
40%-|                                                 s              |-40%
30%-|                                                 U              |-30%
20%-|                                                 U              |-20%
10%-|                                              .  U              |-10%
CPU +65---+70-------+80-------+90-------+100------+110------+120-----+
 Memory
        Physical  PageSpace |         pages/sec  In     Out | FileSystemCache
```

```
-topas_nmon--D=Disk-Stats---------Host=p90n12---------Refresh=2 secs---07:57.40
 CPU-Utilisation-Large-View
100%+s---s---s+--s------+---------+---------+---------+---------+----+100%
90%-|U   U   U   U                                                   |-90%
80%-|U   U   U   U                                                   |-80%
70%-|U   U   U   U                                                   |-70%
60%-|U   U   U   U                                                   |-60%
50%-|U   U   U   U                                                   |-50%
40%-|U   U   U   U                                                   |-40%
30%-|U   U   U   U                                                   |-30%
20%-|U   U   U   U                                                   |-20%
10%-|U.  U   U   U                                                   |-10% .U
CPU +1--------+10-------+20-------+30-------+40-------+50-------+60--+-Avg=LP
100%+-----+---------+---------+---------+---------+---------+--------+100%
90%-|                                                                |-90%
80%-|                                                                |-80%
70%-|                                                                |-70%
60%-|                                                                |-60%
50%-|                                                                |-50%
40%-|                                                                |-40%
30%-|                                                                |-30%
20%-|                                                                |-20%
10%-|                        .                                       |-10%
CPU +65---+70-------+80-------+90-------+100------+110------+120-----+
 Memory
         Physical  PageSpace |         pages/sec  In     Out | FileSystemCache
% Used       8.4%       0.2% | to Paging Space   0.0    0.0 | (numperm)  0.2%
% Free      91.6%      99.8% | to File System    0.0    0.0 | Process    4.7%
GB Used     10.5GB     0.0GB | Page Scans        0.0        | System     3.5%
GB Free    113.8GB    16.0GB | Page Cycles       0.0        | Free      91.6%
Total(GB)  124.2GB    16.0GB | Page Steals       0.0        |          ------
                             | Page Faults   44398.6        | Total    100.0%
------------------------------------------------------------ | numclient  0.2%
Min/Maxperm     3705MB(  3%)  111156MB( 87%) <--% of RAM     | maxclient 87.4%
Min/Maxfree     960   1088      Total Virtual  140.2GB       | User       2.1%
Min/Maxpgahead    2      8   Accessed Virtual   10.1GB  7.2%| Pinned     5.7%
 Top-Processes-(256)-----Mode=3  [1=Basic 2=CPU 3=Perf 4=Size 5=I/O 6=Cmds]--
  PID      %CPU  Size   Res   Res   Res  Char RAM     Paging       Command
           Used    KB   Set  Text  Data   I/O  Use io other repage
 4194994 263.2 1115M 1115M    72 1115M     0  10%    0 43972    0 streammpi.SP
  720930   1.6   448   448     0   448     0   0%    0     0    0 xmgc
```

**Nmon output**

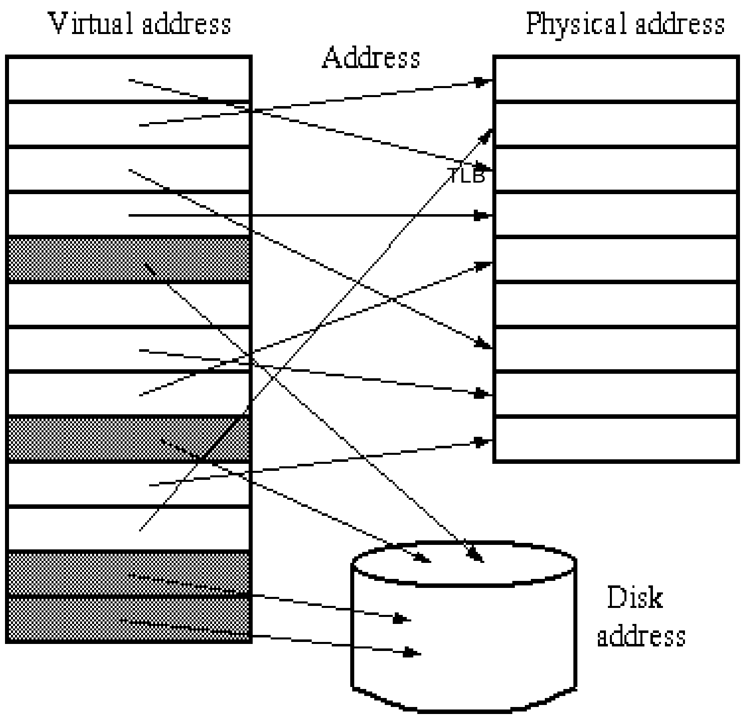
## One POWER7 Core: Tuning Techniques
### Affinities : How to use Memory Affinity

Memory affinity enabled by using:

```
export MEMORY_AFFINITY=MCM
```

# One POWER7 Core: Tuning Techniques
## Memory pages

- Sizes for memory pages on AIX
  - 4 KB aka « Small Pages »
  - 64 KB aka « Medium Pages »
  - 16MB aka « Large Pages »
- Usage of Medium Pages
  - Enhance memory bandwidth
    - Prefetch performance is limited by page size
  - Enhance TLB coverage
    - If memory page information is in TLB: zero cost translations
    - If memory page information is not in TLB: wait for info
- TLB details
  - POWER5+ & POWER6 : 2048 TLB entries
  - POWER7: 512 TLB entries
    - Using Medium Pages allows to cover 32MB of memory vs 2MB with Small Pages
- How to use Medium Pages
  - AIX:
    - Using the "ldedit" command: ldedit -btextpsize=64K -bdatapsize=64K -bstackpsize=64K <exe>
    - Using environment variable LDR_CNTRL=DATAPSIZE=64K@TEXTPSIZE=64K@STACKPSIZE=64K <exe>
  - Linux:
    - 64k memory pages are the default memory pages used by the operating system for latest supported flavors of RHEL and SLES distributions.
    - Nothing to do to use Medium Pages



## How to use Large Pages

- Some HPC applications will benefit from even large pages (16MB)
- Large Pages are pinned and not accessible for data regions that utilize 4KB or 64KB pages

**AIX:**

Need to statically allocate memory pool

```
Ex: vmo -r -o lgpg_regions=300 -o lgpg_size=16777216
```

user must have **CAP_BYPASS_RAC_VMM** and **CAP_PROPAGATE** capabilities

ldedit –blpdata <exe>

Export ......

Issuing the above command will create 300 large pages (4.8GB) dynamically.

**Linux:**

Requires linux kernel version > 2.6.16 + libhugetlbfs.a library

## Monitoring page size usage

```
# vmstat -p all
System configuration: lcpu=16 mem=128000MB
kthr    memory              page              faults        cpu
----- ----------- ------------------------ ------------    -----------
 r  b     avm      fre   re  pi  po  fr   sr  cy  in   sy  cs  us sy id wa
 3  1 1947834 11808993   0   0   0   0    0   0  12 1389 2219  1  0 99  0
  psz     avm     fre  re  pi  po  fr   sr  cy       siz
   4K 1287131 4942320   0   0   0   0    0   0   8463408
  64K   41294  429167   0   0   0   0    0   0    470461
  16M       0    4096   0   0   0   0    0   0      4096

# vmstat -P all
System configuration: mem=128000MB
pgsz            memory                           page
----- ------------------------- ------------------------------------
           siz       avm       fre     re     pi     po     fr      sr     cy
   4K  8463408  1287097  4942194      0      0      0      0       0      0
  64K   470461    40820   429641      0      0      0      0       0      0
  16M     4096        0     4096      0      0      0      0       0      0


# ps -Z -T 90250
   PID    TTY  TIME  DPGSZ SPGSZ TPGSZ SHMPGSZ   CMD
  90250      - 14:02    64K    4K    4K      4K myexe
```

## One POWER7 Core:
## Affinities, SMT and Memory Pages

Goal:

use STREAM_MPI code to show effect of process and thread binding as well as memory affinity on application performance.

| **Binding** | **Memory Affinity** | **Memory Pages** | **Mode** | **STREAM TRIAD (MB/s)** |
|---|---|---|---|---|
| No | No | Small | ST (32 cores) | **??** |
| Yes | No | Small | ST (32 cores) | **??** |
| Yes | Yes | Small | ST (32 cores) | **??** |
| Yes | Yes | Medium | ST (32 cores) | **??** |
| Yes | Yes | Medium | SMT (32 cores – 128 mpi*1thread) | **??** |
| Yes | Yes | Medium | SMT (32 cores – 32 mpi*4thread) | **??** |

## One POWER7 Core:
### Affinities, SMT and Memory Pages

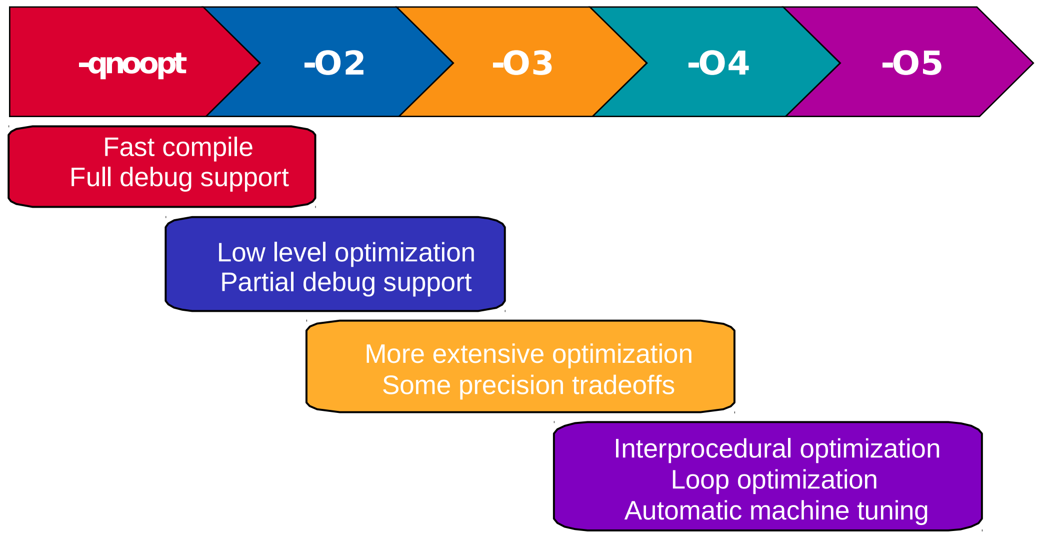
Goal:

use STREAM_MPI code to show effect of process and thread binding as well as memory affinity on application performance.

| Binding | Memory Affinity | Memory Pages | Mode | STREAM TRIAD (MB/s) |
|---|---|---|---|---|
| No | No | Small | ST (32 cores) | **56830** |
| Yes | No | Small | ST (32 cores) | **56564** |
| Yes | Yes | Small | ST (32 cores) | **122409** |
| Yes | Yes | Medium | ST (32 cores) | **122474** |
| Yes | Yes | Medium | SMT (32 cores – 128 mpi*1thread) | **117082** |
| Yes | Yes | Medium | SMT (32 cores – 32 mpi*4thread) | **117065** |

- Binding and memory affinity are mandatory
- SMT doesn’t provide improvement for STREAM
  - Often the case when a resource is overloaded – here memory bandwith
- Medium Pages doesn’t provide significant improvement
  - Was huge on P5/P6 : ~30%
  - Prefetching engines seems to have been improved !

# One POWER7 Core: Tuning Techniques Compiler optimization



- General optimization
  - Unrolling
  - Loop fusion/fission
  - Inlining
  - …
- Compromise between speed and accuracy:
  - O3 –qstrict –qarch=pwr7 –qtune=pwr7

- Compiler Autovectorization
  - For C:
    - **xlc –qarch=pwr7 –qtune=pwr7 –O3 –qhot –qaltivec –qsimd=auto**
  - For Fortran:
    - **xlf –qarch=pwr7 –qtune=pwr7 –O3 –qhot –qsimd=auto**
- May be safer and more efficient if applied only to “vector friendly” routines/instructions blocks,…

- The minimum compiler versions that explicitly support the POWER7 architecture are:
  - **XL C/C++** (C compiler) version **11.1**
  - **XLF** Fortran version **13.1**

## One POWER7 Core: Tuning Techniques
## Tuned Libraries : ESSL Library

ESSL = Engineering and Scientific Subroutine Library

- **Extremely optimized (Serial & SMP)**
- **VMX and VSX enabled for POWER7**
  - ESSL v5.1 includes support for POWER7 unique features.
- **Parallel version**: **PESSL**
- **Compatible BLAS 1, BLAS 2 and BLAS 3**
- **Available on**
  - **POWER5 / POWER6 / POWER7 / BG/L / BG/P / CELL**
  - **AIX / Linux**

More than **450 subroutines** for:

- Linear Algebra Subprograms
- Matrix Operations
- Linear Algebraic Equations
- Eigensystem Analysis
- Fourier Transforms, Convolutions, Correlations and Related Computations
- Sorting and Searching
- Interpolation
- Numerical Quadrature
- Random Number Generation

**http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp**

## One POWER7 Core: Tuning Techniques
## Tuned Libraries : MASS Library

- Mathematical Acceleration Subsystem (MASS)
  - libraries of tuned mathematical intrinsic functions (cos, sin, log,...)

- MASS libraries
  - offer improved performance over the standard mathematical library routines
  - are thread-safe
  - support compilations in C, C++, and Fortran applications.

- Available on
  - POWER5 / POWER6 / POWER7 / BG/L / BG/P / CELL
  - AIX / Linux

- http://www-1.ibm.com/support/docview.wss?rs=2021&context=SSVKBV&uid=swg27005374&loc=en_US&cs=utf-8&lang=en

## One POWER7 Core: Tuning Techniques
## Tuned Libraries : How To Use Libraries

- ESSL
  - Serial:
    - Add **“-lessl”** during link
  - SMP:
    - Add **“-lesslsmp”** during link
    - Add XLFRTEOPTS=parthds=4 at runtime to use 4 threads
  - Tips
    - Use **“-qessl -lessl”** compiler options to transform calls to fortran90 intrinsic function “matmul” into dgemm calls from ESSL.

- MASS
  - Add **“-lmass / -lmassp7 / -lmass_simdp7 / -lmassvp7”** during link
  - An optimized version is included in the compiler libraries (in use with –qhot)
  - Tips
    - Using **“-qhot –lmass”** compiler options allows the compiler to transform loops and generate calls to vector functions from libmass library often very good performance

## One POWER7 Core:
## Compiler optimization / Libraries

- **MMULT**:
  - Matrix multiplication
    - Main routine of TOP500 benchmark (Linpa
  - N = 2000  96MB
  - 2xN**3 operations

```
call
dgemm('N','N',n,n,n,1.0d0,a,n,b,n,0.0d0,c,
n)
c   c=matmul(a,b)
c   call jikloop(n,a,b,c)
```

Goal:

measure the impact of compiler options and use of tuned libraries

## One POWER7 Core:
## Compiler optimization / Libraries

- **MMULT**:
  - Matrix multiplication
    - Main routine of TOP500 benchmark (Linpack)
  - N = 2000  96MB
  - 2xN**3 operations

```
call
dgemm('N','N',n,n,n,1.0d0,a,n,b,n,0.0d0,c,
n)
c   c=matmul(a,b)
c   call jikloop(n,a,b,c)
```

| **MMULT (MFlop/s)** | **-O2** | **-O3** | **-O3 -qhot** | **-O3 –qhot –qessl (ESSL V4.3)** | **-O3 -qhot -qessl –lessl (ESSL V5.1)** |
|---|---|---|---|---|---|
| **Hand coded routine (ijk loop)** | ? | ? | ? | ? | ? |
| **Intrinsic Fortran90 (matmul)** | ? | ? | ? | ? | ? |
| **ESSL library (dgemm) V4.3 / V5.1** | ? | ? | ? | ? | ? |

## One POWER7 Core:
## Compiler optimization / Libraries

- **MMULT**:
  - Matrix multiplication
    - Main routine of TOP500 benchmark (Linpack)
  - N = 2000 → 96MB
  - 2xN**3 operations

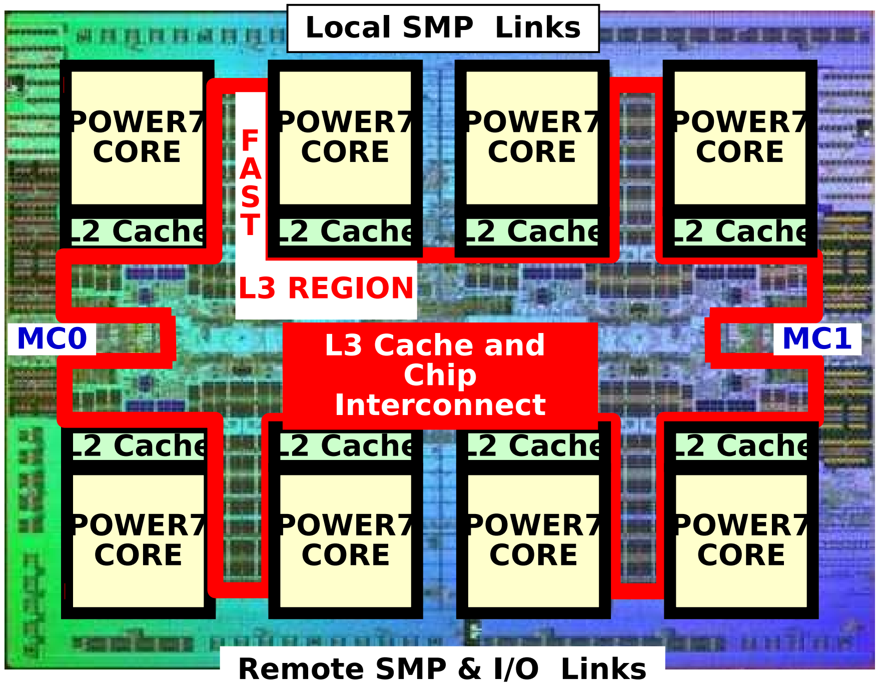
  - Old ESSL (4.3) → **10 GFlop/s** on one core
  - New ESSL (5.1) → **23 GFlop/s** on one core (>87% of peak !)
    - POWER7 tuned version of ESSL
    - VSX version of DGEMM function
    - → more than **2x factor** !!

```
call dgemm('N','N',n,n,n,1.0d0,a,n,b,n,0.0d0,c,n)
c   c=matmul(a,b)
c   call jikloop(n,a,b,c)
```

| **MMULT (MFlop/s)** | **-O2** | **-O3** | **-O3 -qhot** | **-O3 –qhot –qessl (ESSL V4.3)** | **-O3 -qhot -qessl –lessl (ESSL V5.1)** |
|---|---|---|---|---|---|
| **Hand coded routine (ijk loop)** | 525 | 663 | 661 | 661 | 661 |
| **Intrinsic Fortran90 (matmul)** | 5292 | 5292 | 5292 | 9632 | 22865 |
| **ESSL library (dgemm) V4.3 / V5.1** | 9756 / 23175 | 9756 / 23175 | 9756 / 23175 | 9668 | 22985 |

## One POWER7 Processor

## One POWER7 Processor Chip

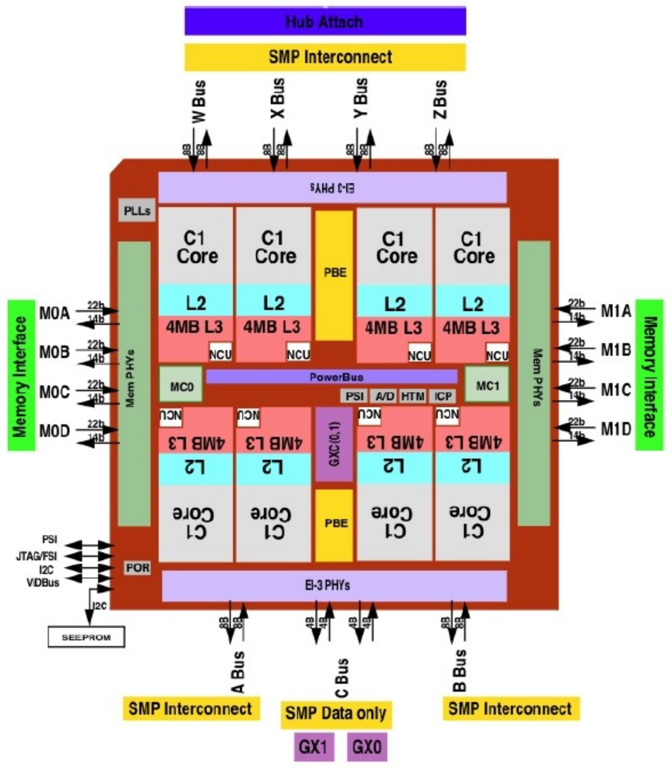


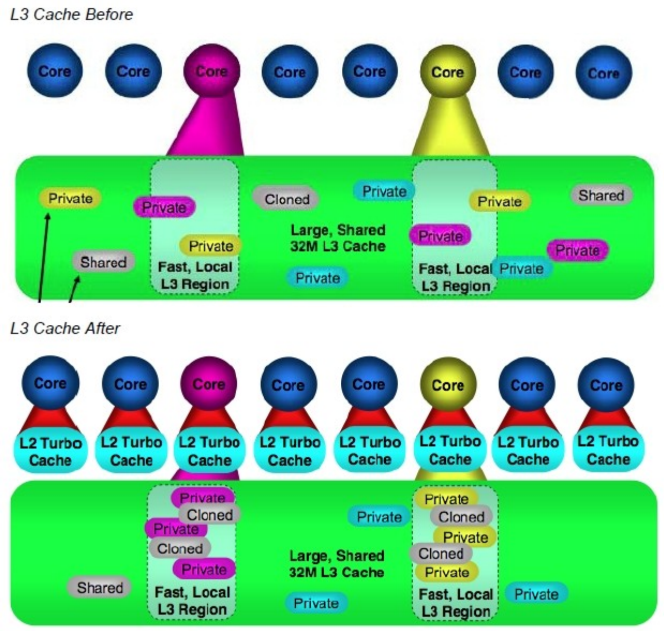
**Binary Compatibility with POWER6**

- Cores : 8 ( 4 / 6 core options )
- 567mm² Technology:
- 45nm lithography, Cu, SOI, eDRAM
- Transistors: 1.2 B
- Equivalent function of 2.7B
- eDRAM efficiency
- Eight processor cores
- 12 execution units per core
- 4 Way SMT per core – up to 4 threads per core
- 32 Threads per chip
- L1: 32 KB I Cache / 32 KB D Cache
- L2: 256 KB per core
- L3: Shared 32MB on chip eDRAM
- Dual DDR3 Memory Controllers
- 100 GB/s Memory bandwidth per chip
- Scalability up to 32 Sockets
- 360 GB/s SMP bandwidth/chip
- 20,000 coherent operations in flight
- Advanced pre-fetching Data and Instructions

IBM

## One POWER7 Processor Chip





| Cache Level | Capacity | Array | Policy | Comment |
|---|---|---|---|---|
| L1 Data | 32K | Fast SRAM | Store-thru | Local thread storage update |
| Private L2 | 256K | Fast SRAM | Store-In | De-coupled global storage update |
| Fast L3 Region | Up to 4M | eDRAM | Partial Victim | Reduced power footprint (up to 4M) |
| Shared L3 | 32M | eDRAM | Adaptive | Large 32M shared footprint |

- The **POWER7 "Fluid" L3 cache structure** provides higher performance by:
  - Automatically cloning shared data to multiple private regions
  - Automatically migrating private footprints (up to 4MB) to the local region (per core) at ~5X lower latency than full L3 caches.
  - Allows a subset of the cores to use the entire L3 cache when some cores are idle

## One POWER7 Processor Chip: L3 Cache

- STREAM MPI on one core, then 8 cores to emphasize L3 performance (local and shared zones)
- So running STREAM on one chip with several configurations:
  - 1 process/thread using L3 <u>local</u> cache (1*2 MB) - 1mpi x 1omp ▯ ?? GB/s
  - 1 process/thread using L3 <u>global</u> cache (1*16 MB) - 1mpi x 1omp ▯ ?? GB/s
  - 1 process/thread using memory (1*128 MB) - 1mpi x 1omp ▯ ?? GB/s
  - 8 processes using all L3 <u>local</u> caches (8*2 MB) - 8mpi x 1omp ▯ ?? GB/s
  - 8 threads using all L3 <u>local</u> caches (8*2 MB) - 1mpi x 8omp ▯ ?? GB/s
  - 8 processes using memory (8*128 MB) - 8mpi x 1omp ▯ ?? GB/s
  - 8 threads using memory (8*128 MB) - 1mpi x 8omp ▯ ?? GB/s

## One POWER7 Processor Chip:
## L3 Cache

- STREAM MPI on one core, then 8 cores to emphasize L3 performance (local and shared zones)
- So running STREAM on one chip with several configurations:
  - 1 process/thread using L3 <u>local</u> cache (1*2 MB) - 1mpi x 1omp ▯ **33** GB/s
  - 1 process/thread using L3 <u>global</u> cache (1*16 MB) - 1mpi x 1omp ▯ **15** GB/s
  - 1 process/thread using memory (1*128 MB) - 1mpi x 1omp ▯ **14** GB/s
  - 8 processes using all L3 <u>local</u> caches (8*2 MB) - 8mpi x 1omp ▯ **248** GB/s
  - 8 threads using all L3 <u>local</u> caches (8*2 MB) - 1mpi x 8omp ▯ **234** GB/s
  - 8 processes using memory (8*128 MB) - 8mpi x 1omp ▯ **31** GB/s
  - 8 threads using memory (8*128 MB) - 1mpi x 8omp ▯ **31** GB/s

## One POWER7 System

# One POWER7 System

## Power 755 : 4-Socket HPC System

4U x 28.8” depth

**approx 10 TFlops per Rack**
**( 10 nodes per Rack )**

ENERGY STAR

AIX 5.3 / 6.1

Linux RHEL / SLES

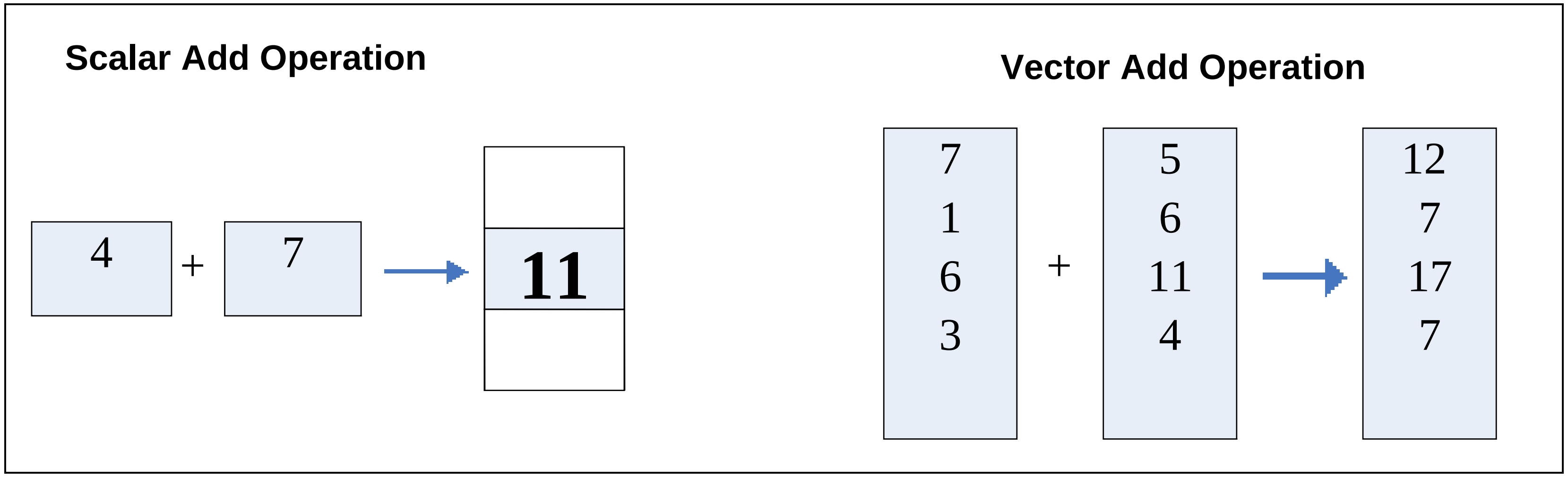
| Power 755 | |
|---|---|
| **POWER7 Architecture** | **4 Processor Sockets = 32 Cores**<br>**8 Core @ 3.3 GHz / 3.6 GHz** |
| **DDR3 Memory** | **128 GB / 256 GB, 32 DIMM Slots** |
| **System Unit SAS SFF Bays** | **Up to 8 disk or SSD**<br>**73 / 146 / 300GB @ 15K (up to 2.4TB)** |
| **System Unit Expansion** | **PCIe x8: 3 Slots (1 shared)**<br>**PCI-X DDR: 2 Slots**<br>**GX++ Bus** |
| **Integrated Ports** | **3 USB, 2 Serial, 2 HMC** |
| **Integrated Ethernet** | **Quad 1Gb Copper**<br>**(Opt: Dual 10Gb Copper or Fiber)** |
| **System Unit Media Bay** | **1 DVD-RAM ( No supported tape bay )** |
| **Cluster** | **Up to 64 nodes**<br>**Ethernet or IB-DDR** |
| **Redundant Power** | **Yes (AC or DC Power)**<br>**Single phase 240vac or -48 VDC** |
| **C** | **NEBS / ETSI for harsh environments** |
| | [illegible] |

## One POWER7 System
### Power 755 Memory Details

| | POWER 5+ 575 (1.9GHz) | Power 575 (4.7GHz) | Power 755 (3.3GHz) |
|---|---|---|---|
| **Latency (cycles/ns)** | 220 cycles / 110ns | 420 cycles / 90ns | 336 cycles / 102ns |
| **Bus** | 2 X DRAM Freq | 4 X DRAM Freq | 6 X DRAM Freq |
| **Memory Controllers per chip** | 1 per chip | 1 | 1 |
| **Peak Bandwidth per chip** | 25GB/s | 34GB/s | 68 GB/s |
| **DRAM Technology** | DDR2 | DDR2 | DDR3 |

## What is “SIMD” on POWER7

- Short vector operations



## Exploiting the VSU

There are 3 ways to exploit the Altivec/VSX instructions available on POWER7

- Auto vectorization with the compiler
- ESSL (and similar SIMD-enabled libraries)
- Hand coding with the compiler

## Language Support

**Compiler Options**

–qarch=pwr7 –qtune=pwr7

–O3 –qhot –qsimd

**Common Vector Intrinsics**

- vec_madd()
- vec_perm (), vec_permi()
- vec_sel()
- vec_splats()
- vec_xlw4(), vec_xld2()

## Using ESSL

- Compiler opts

```
xlc -O3 -q64 -qarch=pwr7-qtune=pwr7
-o dt_fft2d test_fft_d2d.c -lesslsmp –lmass
```

```
/********************************
   test_fft.c
   ***************************/
#include  "essl.h"
#define DIMENSION 2
…
 dcmplx  *aa, *bb, *cc;
  dcmplx  **xx, **yy, **zz;
  double  *aux1, *aux2;
  int naux1,naux2;
  int N[DIMENSION],
   incx[DIMENSION],
   incy[DIMENSION],
  isign[DIMENSION]={1,1};
```

```
  N[0] = nn;
  N[1] = nn;
  incx[0] = 1;
  incy[0] = 1;
… stride(N[1],N[0],&(incy[1]),"Z",0);
  incx[1] = incy[1];
  incmxx=nn*incx[1];
  incmyy=nn*incy[1];
…
dcftd (1, 2, xx, incx, incmxx, yy,
   incy, incmyy, N, m, isign, 1.0 ,
   aux1, naux1, aux2, naux2);
dcftd (0, 2, xx, incx, incmxx, yy,
   incy, incmyy, N, m, isign, 1.0,
   aux1, naux1, aux2, naux2);
```

## Handcoding

Scalar

```
a = (float *) malloc(arraylen*sizeof(float));
b = (float *) malloc(arraylen*sizeof(float));
for (i=0; i<arraylen; i++) {
  b[i] = alpha*a[i] + b[i];
}
```

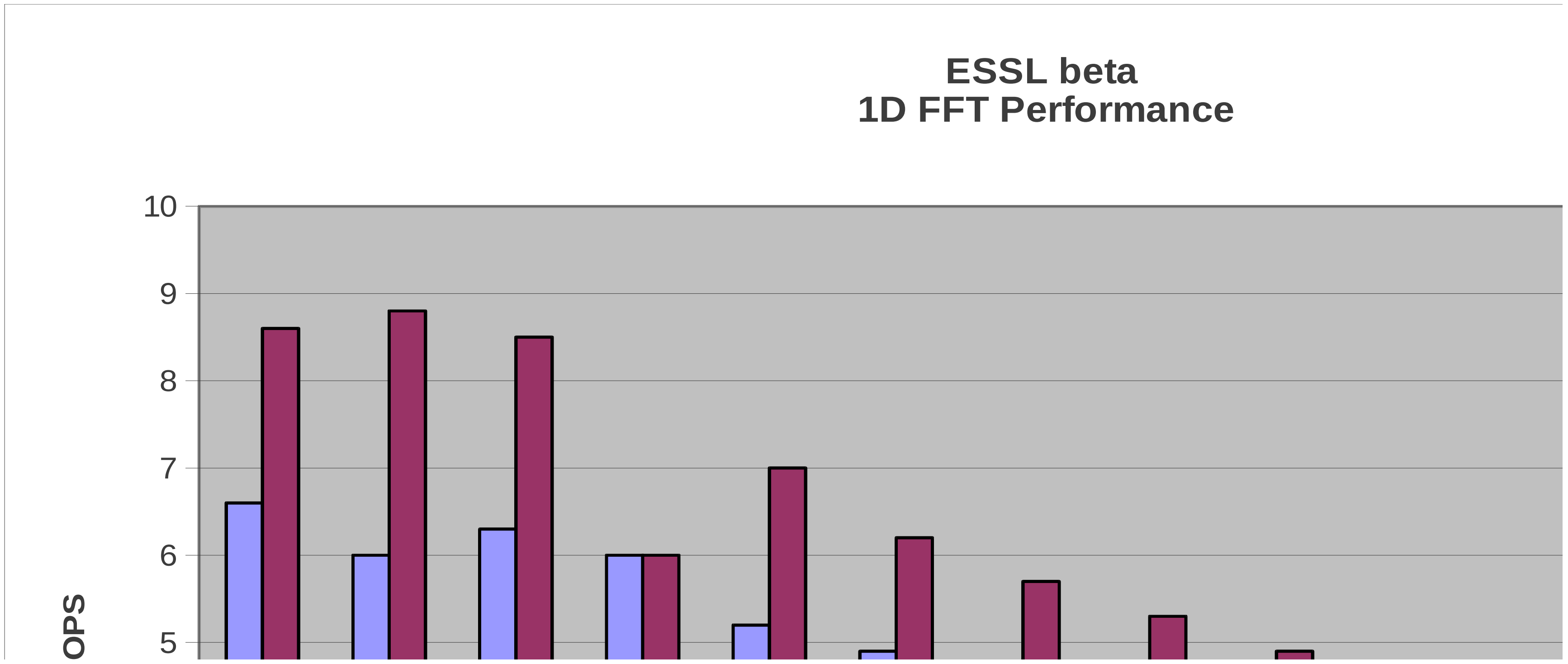
Altivec

```
a = (float *) vec_malloc(arraylen*sizeof(float));
c = (float *) vec_malloc(arraylen*sizeof(float));
vAlpha = vec_splats(alpha);
for (i=0; i<arraylen; i+=4) {
  vA = (vector float *) &(a[i]);
  vC = (vector float *) &(c[i]);
  *vC = vec_madd(vAlpha,*vA,*vC);
}
```

VSX

```
a = (double *)
      vec_malloc(arraylen*sizeof(double));
b = (double *)
      vec_malloc(arraylen*sizeof(double));
vAlpha = vec_splats(alpha);
for (int i=0; i<arraylen; i+=4) {
    vA1 = (vector double*) &(a[i]);
    vB1 = (vector double*) &(b[i]);
    *vB1 = vec_madd(vAlpha,*vA1,*vB1);
    vA2 = (vector double*) &(a[i+2]);
    vB2 = (vector double*) &(b[i+2]);
    *vB2 = vec_madd(vAlpha,*vA2,*vB2);
}
```

# Scalar vs VSX - ESSL (1D) performance



**These are preliminary data only. ESSL release performance may be different**

## One POWER7 Processor Chip: VSX

- Sample code with fmadd
- running with several configurations:
  - Double data type (64 bit) with vsx enabled (vector)  **??** seconds
  - Double data type (64 bit) with vsx disabled (scalar)  **??** Seconds

  - FLOAT data type (32 bit) with vsx enabled (vector)  **??** seconds
  - FLOAT data type (32 bit) with vsx disabled (scalar)  **??** seconds

## One POWER7 Processor Chip: VSX

- Sample code with fmadd
- running with several configurations:
  - Double data type (64 bit) with vsx enabled (vector)  **15.0** seconds
  - Double data type (64 bit) with vsx disabled (scalar)  **23.2** seconds

  - FLOAT data type (32 bit) with vsx enabled (vector)  **7.8** seconds
  - FLOAT data type (32 bit) with vsx disabled (scalar)  **22.6** seconds

## Facilitate I/O operations

Use fast filesystem
- Parallel file-system like GPFS if running on several nodes (clusters, BlueGene)
- Build on local disks if running on one node
- Forget NFS

Use MPI/IO features of MPI2
- Parallel reads/writes to files from several nodes
  - All MPI tasks are reading/writing at the same time on the same file

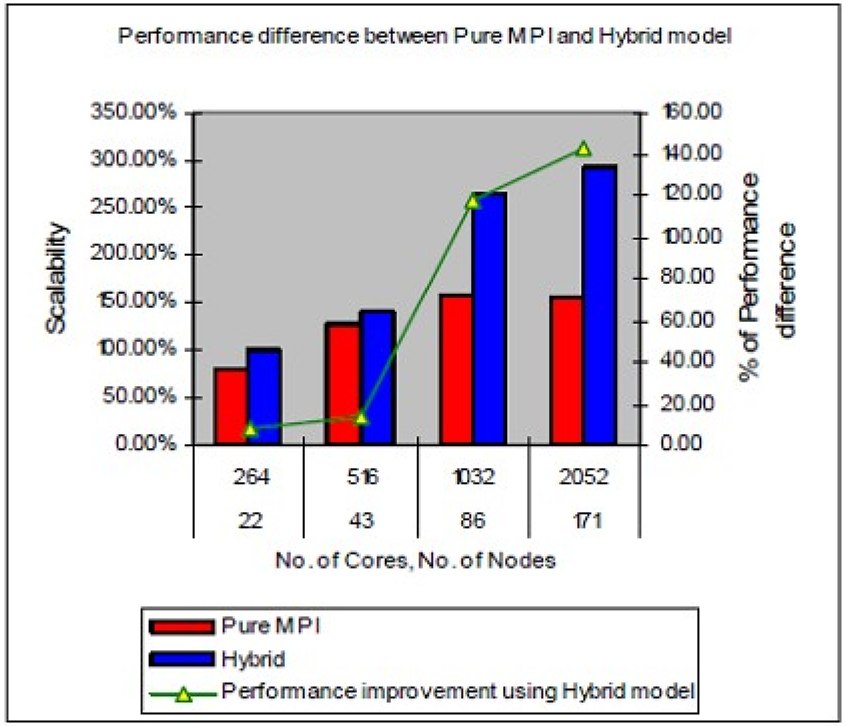
Try to use asynchronous I/O
- Overlap I/O operations with computation or communication

Try to use memory instead of disks
- Fill in the memory before writing to disks
  - File mapping, manual management of I/O in source code,…

## Improve your application performance & scalability

### Using Hybrid (MPI + OpenMP) parallelization

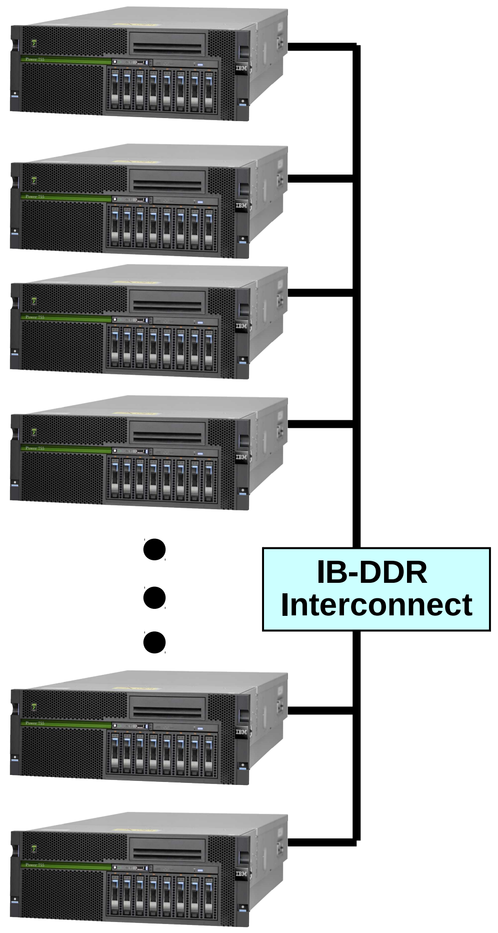
**What is CP2K?**
CP2K is a freely available (GPL) program, written in Fortran 95, to perform atomistic and molecular simulations of **solid state, liquid, molecular and biological systems**.



1. Hyper-threading is enabled and hence double the number of cores
2. Physical cores are mostly used to run the MPI process and logical cores are only used for running OpenMP threads.
3. Super linear scalability was obtained using hybrid parallelization by 17.46% and 42.94% (117% and 142% total improvement compared with pure MPI) for 1032 and 2052 cores respectively.

## One POWER7 Cluster

# One POWER7 Cluster
## Power 755 HPC Cluster Node



Data Center in a Rack
Up to 10 Nodes per Rack
Air cooled

| | 1H / 2010 |
|---|---|
| Scaling | 64 nodes (32 Cores/node)<br>54 TFlops |
| Operating Systems | AIX 6.1 TL 04 / 05<br>Linux |
| HPC Stack Levels | xCAT v2.3.x<br>GPFS v3.3.x<br>PESSL v3.3.x<br>LL v4.1.x<br>PE v5.2.x |

## Profiling

Gprof/prof

- Produce application profiles showing time spent in subroutines and how many times they were called
- Gprof also displays call graph information
- Application instrumentation adds some overhead to profiling run

Tprof

- Uses AIX trace facility to profile your application
- Shows time spent in application subroutines, shared libraries, other system activity
- Can also microprofile: displays ticks for each source line

Mpitrace

- Measures time spent in MPI routines
- Shows message sizes used

## Gprof/prof

Compile and link with `-pg` for gprof, `-p` for prof

  Also recommend using `-g` and `–qfullpath`

Run program normally

  Gprof will generate `gmon.taskid.out` files at end of run

  Prof will generate `mon.taskid.out` files at end of run

Generate profile output:

```
gprof <executable> gmon.0.out gmon.1.out ...
prof –m mon.0.out mon.1.out ...
```

Can also visualize gprof profile with Xprofiler

```
Xprofiler <executable> gmon.0.out gmon.1.out ...
```

## Gprof example output: call graph

```
 called/total       parents
index  %time    self  descendents  called+self    name           index
                                   called/total       children

                0.00       540.42       1/1           .__start [2]
[1]      68.3   0.00       540.42       1         .main [1]
                0.00       538.97       1/1           .EvolveHierarchy__FR14HierarchyEntryR11TopGridDataP16...PP19LevelHierarchyEntryd [3]
                0.00         1.45       1/1           .InitializeNew__FPcR14HierarchyEntryR11TopGridDataR16ExternalBoundaryPd [80]
                0.00         0.00       1/1           .CommunicationInitialize__FPiPPPc [252]
                0.00         0.00       1/1           .__ct__16ExternalBoundaryFv [264]
                0.00         0.00       1/1           .InterpretCommandLine__FiPPcPcRiN74T2PiT13_PdT15_T4T1 [256]
                0.00         0.00       1/1           .my_exit__Fi [266]
                0.00         0.00       1/1           .AddLevel__FPP19LevelHierarchyEntryP14HierarchyEntryi [250]

-----------------------------------------------

6.6s                                                 <spontaneous>
[2]      68.3   0.00       540.42                 .__start [2]
                0.00       540.42       1/1           .main [1]

-----------------------------------------------

                0.00       538.97       1/1           .main [1]
[3]      68.1   0.00       538.97       1         .EvolveHierarchy__FR14HierarchyEntryR11TopGridDataP16...PP19LevelHierarchyEntryd [3]
                0.00       463.54      30/30          .EvolveLevel__FP11TopGridDataPP19LevelHierarchyEntryidP16ExternalBoundary [4]
                0.01        61.31      31/31          .RebuildHierarchy__FP11TopGridDataPP19LevelHierarchyEntryi [16]
                4.61         9.37    3840/3840        .ComputeTimeStep__4gridFv [35]
                0.00         0.07      64/3904        .SetExternalBoundaryValues__4gridFP16ExternalBoundary [59]
                0.00         0.05      64/64          .CopyOverlappingZones__FP4gridP11TopGridDataPP19LevelHierarchyEntryi [123]
                0.00         0.01      30/30          .OutputLevelInformation__FP4FILER11TopGridDataPP19LevelHierarchyEntry [157]
                0.00         0.00      31/31          .CheckForOutput__FP14HierarchyEntryR11TopGridDataP16ExternalBoundaryRi [242]
                0.00         0.00      30/30          .CommunicationMinValue__Fd [244]
                0.00         0.00      30/30          .CheckForTimeAction__FPP19LevelHierarchyEntryR11TopGridData [243]
                0.00         0.00      30/545         .CosmologyComputeExpansionFactor__FdPdT2 [226]
                0.00         0.00       2/74806       .ReturnCPUTime__Fv [216]

-----------------------------------------------
```

...

# Gprof example output: call graph (cont.)

```
                                                        . . .

                0.01        0.00      30/30492       .PrepareFFT__4gridFP6regioniPi [152]
                0.01        0.00      30/30492       .FinishFFT__4gridFP6regioniPi [135]
                2.90        0.00    6720/30492       .CommunicationTranspose__FP6regioniT1N22 [47]
               10.25        0.00   23712/30492       .CommunicationSendRegion__4gridFP4gridiN22PiT5 [45]
[38]     1.7   13.18        0.00   30492         .copy3d [38]

-----------------------------------------------

                0.25        2.36  122880/614400      .s90_1d [71]
                1.02        9.43  491520/614400      .s90_2d [42]
[39]     1.7    1.27       11.79  614400         .sf90 [39]
                0.94       10.85  614400/614400      .__singleton_NMOD_fftn [43]

-----------------------------------------------

                0.00       12.87    1920/1920        .IPRA.$EvolveLevel__FP11TopGridDataPP19...EntryidP16ExternalBoundary [5]
[40]     1.6    0.00       12.87    1920         .UpdateParticlePositions__FP4grid [40]
                8.03        0.00    1920/2040        .UpdateParticlePosition__4gridFd [48]
                4.84        0.00    3840/3840        .UpdateParticleVelocity__4gridFd [56]
                0.00        0.00    5760/74969       .DebugCheck__4gridFPc [153]

-----------------------------------------------

                0.00        0.05    4096/1110016     .CopyOverlappingZones__FP4gridP11TopGridDataPP19LevelHierarchyEntryi [123]
                0.00        1.42  122880/1110016     .ComputePotentialFieldLevelZero__FP11TopGridDataPP14HierarchyEntryi [23]
                0.01        2.84  245760/1110016     .PrepareDensityField__FPP19LevelHierarchyEntryiP11TopGridData [19]
                0.01        2.84  245760/1110016     .CopyOverlappingParticleMassFields__FP4gridP11TopGridDataPP19...Entryi [69]
                0.01        5.68  491520/1110016     .SetBoundaryConditions__FPP14...P11TopGridDataP16ExternalBoundary [46]
[41]     1.6    0.03       12.82 1110016         .CheckForOverlap__4gridFP4gridP13boundary_typeT2M4gridFP4gridPd_i [41]
                1.45        4.14   43859/43923       .CopyZonesFromGrid__4gridFP4gridPd [52]
                0.90        2.04   21600/21600       .AddOverlappingParticleMassField__4gridFP4gridPd [68]
                0.14        2.04   21570/21570       .CopyPotentialField__4gridFP4gridPd [76]
                0.08        2.04   21570/21570       .CopyOverlappingMassField__4gridFP4gridPd [77]
                0.00        0.00   38354/38508       .ReportMemoryUsage__FPc [217]

                                                         . . .
```

## Gprof example output: flat profile

```
 %   cumulative   self              self     total
time   seconds   seconds    calls  ms/call  ms/call  name
11.8      94.67    94.67                             ._sqrt [11]
 9.7     172.39    77.72     4020    19.33    40.96  .yeuler_sweep [8]
 8.6     241.69    69.30    12060     5.75     5.75  .euler [13]
 8.3     308.28    66.59                             ._stripe_hal_newpkts__FUiP11hal_param_t [14]
 8.0     372.29    64.01  9696240     0.01     0.01  .intvar [15]
 7.7     434.46    62.17    12060     5.16    10.46  .inteuler [9]
 7.6     495.76    61.30      262   233.97   233.97  .CommunicationTransferParticles__4gridFPP4gridiPiT3PP9float_intT2 [18]
 5.4     539.12    43.36    12060     3.60     3.60  .twoshock [20]
 4.6     575.67    36.55                             ._lapi_dispatcher__FUib [21]
 4.2     609.27    33.60                             ._lapi_shm_dispatcher__FUi [22]
 2.7     631.19    21.92    24120     0.91     0.91  .pgas2d_dual [25]
 1.9     646.50    15.31       90   170.11   170.11  .cic_interp [31]
 1.7     659.97    13.47       30   449.00   449.00  .dep_grid_cic [37]
 1.6     673.15    13.18    30492     0.43     0.43  .copy3d [38]
 1.5     685.48    12.33     4020     3.07    24.69  .zeuler_sweep [10]
 1.1     694.01     8.53     2040     4.18     4.18  .UpdateParticlePosition__4gridFd [48]
 0.9     701.42     7.41       60   123.50   123.50  .calc_dt [50]
 0.9     708.27     6.85   614400     0.01     0.01  .transform [51]
 0.7     713.61     5.34                             .LAPI__Msgpoll [53]
 0.6     718.70     5.09       30   169.67   169.67  .expand_terms [55]
 0.6     723.54     4.84     3840     1.26     1.26  .UpdateParticleVelocity__4gridFd [56]
 0.6     728.15     4.61     3840     1.20     3.64  .ComputeTimeStep__4gridFv [35]
 0.5     732.39     4.24      366    11.58    11.58  .SetExternalBoundary__16ExternalBoundaryFiPiN32PdT1 [60]
 0.5     736.63     4.24       30   141.33   141.33  .cic_deposit [62]
 0.5     740.44     3.81   614400     0.01     0.01  .permute [63]
 0.4     744.01     3.57     4020     0.89    22.51  .xeuler_sweep [12]
 0.4     747.43     3.42       30   114.00   114.00  .CopyBaryonFieldToOldBaryonField__4gridFv@AF2_1 [65]
 0.4     750.73     3.30     2400     1.38     1.38  .copy3drt [66]
 0.4     753.67     2.94       90    32.67    32.67  .ComputePressureDualEnergyFormalism__4gridFdPd [67]
 0.4     756.57     2.90       60    48.33    87.50  .ComputeAccelerationField__4gridFiT1 [54]
 0.3     759.29     2.72                             .reduce_onnode_bin_pairwise [70]
```

**Turn up optimization to eliminate** (._sqrt [11])

**Candidate for inlining** (.intvar [15])

. . .

## Tprof

Flat Profile Mode

```
tprof –usz –p binary_name –x poe path_to_binary_name ...
```

Samples application, kernel (on high level) and shared library information
Uses AIX trace facility —  only one trace per OS image possible at same time
Produces `poe.prof` file

Microprofiling

```
tprof -usz -M source_path_list -m object_list -L object_list -p
  process_list -x poe path_to_binary_name ...
```

`-M`: path to source files
`-m`: enable microprofiling for listed executables, shared libraries
`-L`: annotate listing files of which executables and shared libraries
`-p`: which processes to profile
In addition, produces `poe.source_file.mprof` files

## Tprof Example: poe.prof file

```
Configuration information
=========================
System: AIX 5.3 Node: v60n03 Machine: 00C1D5D24C00
Tprof command was:
    tprof -usz -M /vol/xcae1/rrajan/amr_mpi/src -m /gpfs/fs1/tmp/rrajan/bin/enzo
 -L /gpfs/fs1/tmp/rrajan/bin/enzo -p enzo -x poe /gpfs/fs1//tmp/rrajan/bin/enzo
 -d param_large.dat -procs 128 -hfile host.list
Trace command was:
    /usr/bin/trace -ad -M -L 829432627 -T 500000 -j 000,00A,001,002,003,38F,005,006,134,139,5A2,5A5,465,234, -o -
Total Samples = 1789147
Traced Time = 338.14s (out of a total execution time of 338.14s)
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

Process                               FREQ  Total Kernel   User Shared  Other
=======                               ====  ===== ======   ==== ======  =====
wait                                    64 982264 982264      0      0      0
/gpfs/fs1/tmp/rrajan/bin/enzo          158 803026  53376 558683 190966      1
/etc/pmdv4                              33   1326   1294      0     32      0
/usr/bin/tprof                           1    557    187     92    278      0
mmfsd64                                 67    424    380     37      7      0
sshd:                                    3    301    238     32     31      0
CqKp                                     7    243    243      0      0      0
dog                                      7    195    195      0      0      0
/usr/bin/tee                             1    194    188      4      2      0
/usr/bin/poe                             1    128     98      4     26      0
swapper                                  4    120    120      0      0      0
/usr/lpp/LoadL/full/bin/LoadL_startd     8     81     70      0     11      0
/usr/sbin/syncd                          3     66     64      0      2      0
/usr/java14/jre/bin/java                 3     45      1      0     40      4
hats_nim                                28     40     27      7      6      0
IBM.ConfigRMd                            2     23     22      1      0      0

                                                      . . .
```

# Tprof Example: poe.prof file (cont.)

```
                                        . . .

 Total Ticks For All Processes (USER) = 558867

User Process                                         Ticks    %    Address  Bytes
=============                                        ===== ====== ======= =====
/gpfs/fs1/tmp/rrajan/bin/enzo                       558684  31.23 100000288 2dd3d4
/usr/bin/tprof                                          92   0.01 10000100  2ce98
mmfsd64                                                 36   0.00 10001f8e0 3132b1
sshd:                                                   31   0.00 1000afb8  351bd
hats_nim                                                 7   0.00 100a4cb8  b2785
/usr/bin/tee                                             4   0.00 10000100    8f0
/usr/bin/poe                                             4   0.00 10000150  47d7d
/usr/lpp/LoadL/full/bin/LoadL_kbdd                       3   0.00 10000100  1f5c5
/usr/bin/vi                                              3   0.00 10000100  3f7a8
/usr/bin/view                                            2   0.00 10000100  3f7a8
/bin/ls                                                  1   0.00 10000100   5584

  Profile: /gpfs/fs1/tmp/rrajan/bin/enzo

  Total Ticks For All Processes (/gpfs/fs1/tmp/rrajan/bin/enzo) = 558684

Subroutine                   Ticks    %   Source                Address  Bytes
==========                   ===== ====== ======                ======= =====
.euler                       90131   5.04 /amr_mpi/src/euler.f   23f6f8   54e0
.inteuler                    80434   4.50 r_mpi/src/inteuler.f   2363b8   5a20
.twoshock                    64594   3.61 r_mpi/src/twoshock.f   221cd8   1f00
.intvar                      58007   3.24 amr_mpi/src/intvar.f   23bdd8   26e0
nt*,int*,float_int**,int)    46045   2.57 nTransferParticles.C   19dcd8   2240
.pgas2d_dual                 34543   1.93 pi/src/pgas2d_dual.f   21f578    620
.yeuler_sweep                30193   1.69 i/src/yeuler_sweep.f   244bd8   1a00
.dep_grid_cic                18280   1.02 r_mpi/src/grid_cic.f   1d6bd8   3a80
.cic_interp                  18232   1.02 mpi/src/cic_interp.f   1e9938   10e0
.zeuler_sweep                13679   0.76 i/src/zeuler_sweep.f   2465d8   1ac0
.copy3d                      11779   0.66 _mpi/src/utilities.f     9878    3a0
.calc_dt                     11024   0.62 mr_mpi/src/calc_dt.f   1c94b8   2600

                                        . . .
```

## Microprofiling file: poe.euler.f.mprof (cont.)

```
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

        Total Ticks for .euler = 90131

  Line  Ticks Source

   136      3 c
   137      -       do i=1,6
   138      -          lWarn(i) = .FALSE.
   139      8          iWarnCount(i) = 0
   140      -       enddo
   141      - c
   142      - c     Set constants
   143      - c
   144      -       qa = (gamma + 1.0)/(2.0*gamma)
   145      -       qb = (gamma - 1.0)/(gamma + 1.0)
   146      - c
   147      - c     Loop over sweep lines (in this slice)
   148      - c
   149     10       do j=j1, j2
   150      - c
   151      - c     Evaluate time-averaged quantities
   152      - c     (see Colella, Siam J Sci Stat Comput 1982, 3, 77.  Appendix)
   153      - c
   154   2158          do i=i1, i2+1
   155      - c
   156    266             sn(i)    = sign(1.0, -ubar(i,j))
   157      - c
   158      - c     Collect values of interest depending on which way fluid is flowing
   159      - c
   160   5794             if (sn(i) .lt. 0.0) then
   161    233                u0(i) = uls(i,j)
   162    128                p0(i) = pls(i,j)
   163    510                d0(i) = dls(i,j)
   164      - c     vb(i) = vls(i,j)
   165      - c     wb(i) = wls(i,j)
   166      - c     geb(i) = gels(i,j)
   167     10             else
   168    225                u0(i) = urs(i,j)
   169    112                p0(i) = prs(i,j)
   170    544                d0(i) = drs(i,j)
                                        . . .
```

## Mpitrace

Link with libmpitrace

- Simply add `–L$(IHPCT_BASE)/lib –lmpitrace -llicense` to your linker options

Some output control with environment variables

- `SAVE_ALL_TASKS`
  - write info for all tasks or just {0,min,max,median}
- `TRACE_ALL_EVENTS`
  - write time event trace (for PeekPerf)

Instrument code sections

- Use `trace_start()`/`trace_stop()` to demarcate sections in your code where you want trace information collected
- User `summary_start()`/`summary_stop()` to demarcate sections in your code where you want to collect MPI statistics
- Useful for getting rid of program initialization/finalization times

# Mpitrace Example Output: mpi_profile.0

```
Data for MPI rank 0 of 64:
Times and statistics from summary_start() to summary_stop().
-----------------------------------------------------------------
MPI Routine                  #calls     avg. bytes      time(sec)
-----------------------------------------------------------------
MPI_Send                        126         6868.8          0.001
MPI_Isend                      3926       377093.1          0.244
MPI_Recv                       4073       364069.5         28.514
MPI_Sendrecv                  11340       254168.7         23.523
MPI_Test                       4635            0.0          0.023
MPI_Allgather                   133            4.0          6.251
MPI_Allgatherv                  133          104.0          0.066
MPI_Allreduce                    30            8.0          0.554
-----------------------------------------------------------------
MPI task 0 of 64 had the minimum communication time.
total communication time = 59.174 seconds.
total elapsed time       = 519.055 seconds.
user cpu time            = 256.401 seconds.
system time              = 0.082 seconds.
maximum memory size      = 781664 KBytes.

-----------------------------------------------------------------
Message size distributions:

MPI_Send                  #calls    avg. bytes      time(sec)
                               5          64.0          0.000
                               1         128.0          0.000
                               6         906.7          0.000
                              49        1592.2          0.000
                               5        2304.0          0.000
                              13        7256.6          0.000
                              29       12219.6          0.000
                              18       17852.4          0.000

                                   . . .
```

## Mpitrace Example Output: mpi_profile.0 (cont.)

```
                                   . . .

MPI_Recv                  #calls    avg. bytes      time(sec)
                               7          64.0          0.000
                               8         128.0          0.001
                               7         219.4          0.000
                               4         400.0          0.000
                               1         960.0          0.000
                             968        1510.2          2.625
                             246        5869.5          0.002
                              43       13911.8          0.001
                              67       23889.2          0.009
                            1456       46124.9          8.076
                             360       86832.0          0.024
                             360      786432.0         12.810
                             180     1292832.0          0.359
                             366     2359296.0          4.606

MPI_Sendrecv              #calls    avg. bytes      time(sec)
                            5760           0.0         13.930
                            3750      262144.0          4.620
                              30      327680.0          0.081
                            1680     1048576.0          4.483
                             120     1064960.0          0.409


MPI_Allgather             #calls    avg. bytes      time(sec)
                             133           4.0          6.251

MPI_Allgatherv            #calls    avg. bytes      time(sec)
                             133         104.0          0.066

MPI_Allreduce             #calls    avg. bytes      time(sec)
                              30           8.0          0.554
----------------------------------------------------------------
-
```

## Mpitrace Example Output: mpi_profile.0 (cont.)

```
Communication summary for all tasks:

  minimum communication time = 59.174 sec for task 0
  median  communication time = 65.827 sec for task 39
  maximum communication time = 72.788 sec for task 17


MPI tasks sorted by communication time:
taskid      comm(s)  elapsed(s)     user(s)     size(KB)   switches
     0        59.17       519.06      256.40       781664       21192
    32        60.43       519.05      256.43       741088       29896
    60        60.62       519.05      257.54       774176       21030
     4        61.03       519.06      257.87       782816       19270
     6        61.38       519.06      258.47       802592       21152
    34        63.00       519.06      258.50       775776       22381
    20        63.01       519.06      258.09       794848       22661
    12        63.12       519.06      257.88       739104       25286
    56        63.43       519.05      258.00       782560       20430
    44        63.59       519.03      257.85       739680       25790
    22        63.83       519.06      258.38       784160       20038
     2        63.84       519.04      258.91       739808       20662
    61        63.96       519.05      260.10       777952       20463
                              . . .
    27        71.09       519.06      259.52       740704       20958
    63        71.45       519.06      259.51       753312       31524
    42        71.48       519.06      258.32       747552       31142
    41        72.18       519.05      260.10       733984       20544
    43        72.21       519.05      259.51       753440       22154
    18        72.63       519.05      259.16       751648       19644
    17        72.79       519.05      260.15       750752       25296
```

## Conclusion

- POWER7 is a very efficient processor for HPC applications but a few basic techniques are required to take advantage of the whole power
  - Enforce Affinities for process, threads and memory
  - Enable Vectorization thru compiler options or tuned libraries
  - Use Tuned Libraries ESSL & MASS
  - Optimize throughput with SMT capabilities
  - Enhance memory subsystem performance with Medium Pages usage
  - Explore the opportunity to adopt Hybrid model
- These recommendations are valid for all POWER7 based HPC systems.
- Most of these “tricks” can be configured in the environment for a comfortable user experience.

# Thank you...