Intel® Xeon 5500 Platforms, Integrated Memory Controllers and NUMA

David Levinthal, Julia Fedorova, Dmitry Ryabtsev
SSG/DPD/PAT

Agenda

NUMA and Enabling: Overview

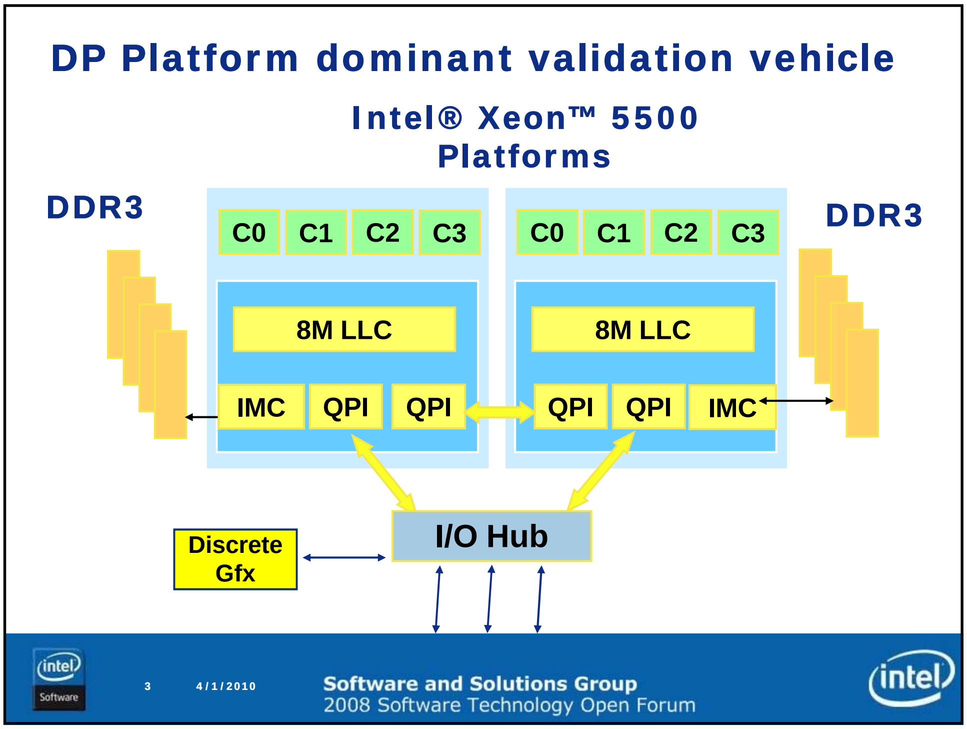
Topology Overview

BIOS Options

OS dependent NUMA concerns

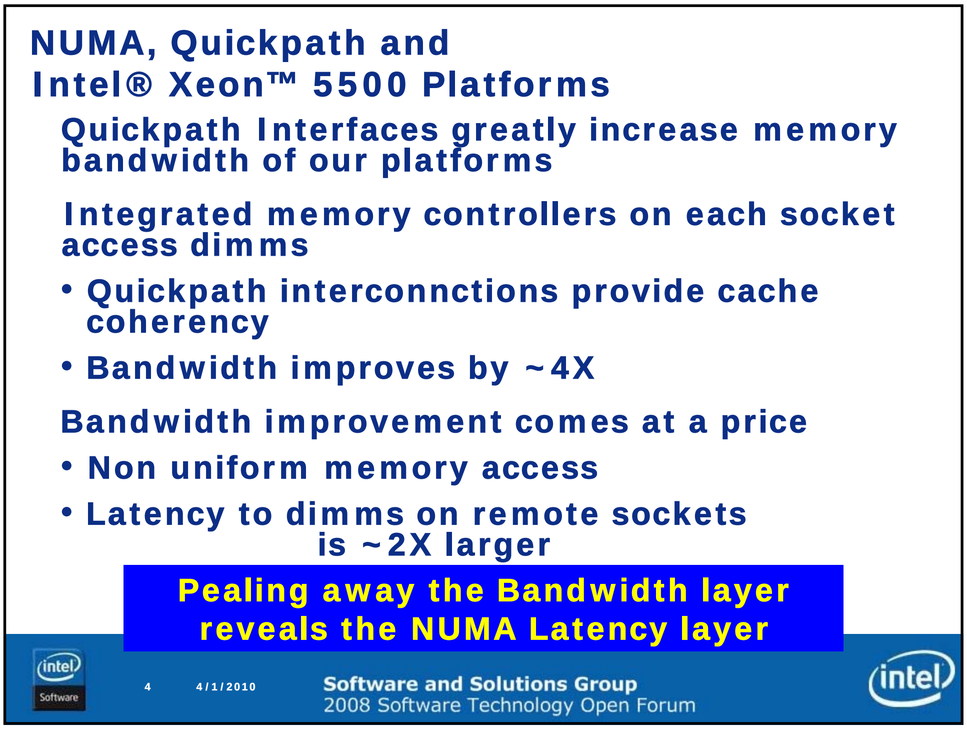
Identifying memory locality (and lack thereof) on Intel® Xeon 5500 processors

Summary



DP Platform dominant validation vehicle

Intel® Xeon™ 5500 Platforms



NUMA, Quickpath and Intel® Xeon™ 5500 Platforms

Quickpath Interfaces greatly increase memory bandwidth of our platforms

Integrated memory controllers on each socket access dimms

- Quickpath interconnctions provide cache coherency
- Bandwidth improves by ~4X

Bandwidth improvement comes at a price

- Non uniform memory access
- Latency to dimms on remote sockets is ~2X larger

Pealing away the Bandwidth layer reveals the NUMA Latency layer

NUMA Modes on DP Systems Controlled in BIOS

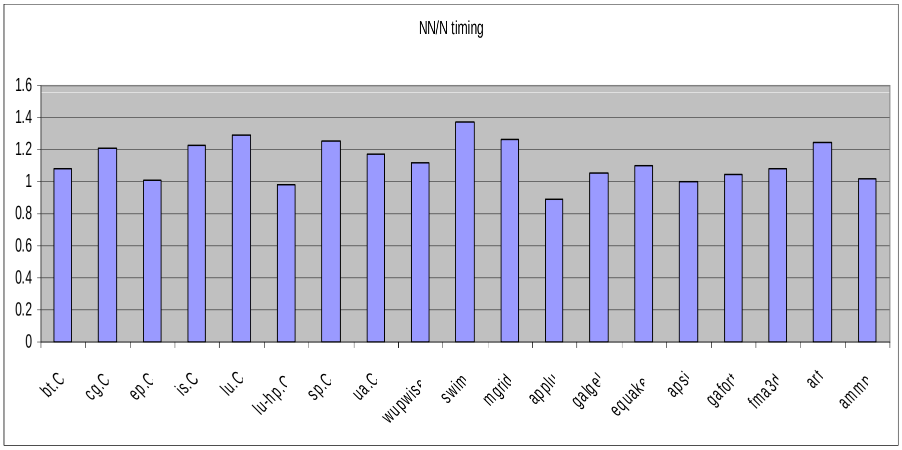
Non Numa

- Even/Odd lines assigned to sockets 0/1
  - Line interleaving

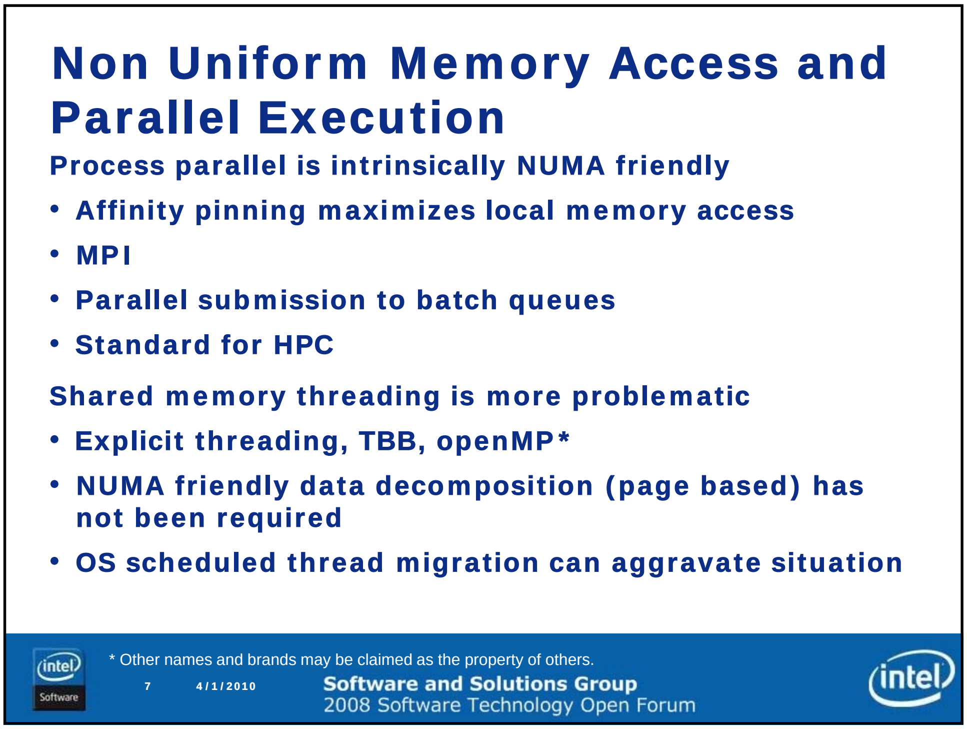
NUMA mode

- First Half of memory space on socket 0
- Second half on socket 1
- Default on Intel® Xeon™ 5500 Processors

NON-NUMA/NUMA Timings for Specomp* and NAS* Parallel Benchmarks



* Other names and brands may be claimed as the property of others.



Non Uniform Memory Access and Parallel Execution

Process parallel is intrinsically NUMA friendly

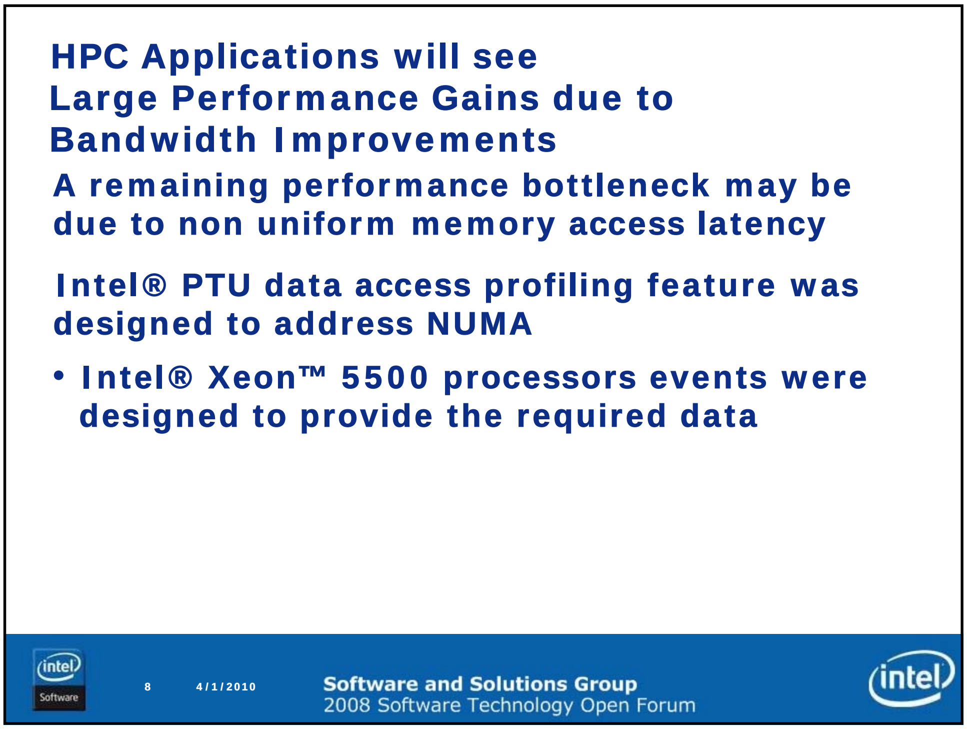
- Affinity pinning maximizes local memory access
- MPI
- Parallel submission to batch queues
- Standard for HPC

Shared memory threading is more problematic

- Explicit threading, TBB, openMP*
- NUMA friendly data decomposition (page based) has not been required
- OS scheduled thread migration can aggravate situation

* Other names and brands may be claimed as the property of others.

7 4/1/2010 Software and Solutions Group
2008 Software Technology Open Forum



HPC Applications will see Large Performance Gains due to Bandwidth Improvements

A remaining performance bottleneck may be due to non uniform memory access latency

Intel® PTU data access profiling feature was designed to address NUMA

- Intel® Xeon™ 5500 processors events were designed to provide the required data

8 4/1/2010 Software and Solutions Group
2008 Software Technology Open Forum

Data Access Events on Intel® Xeon™ 5500 processors Reveal NUMA Access Pattern

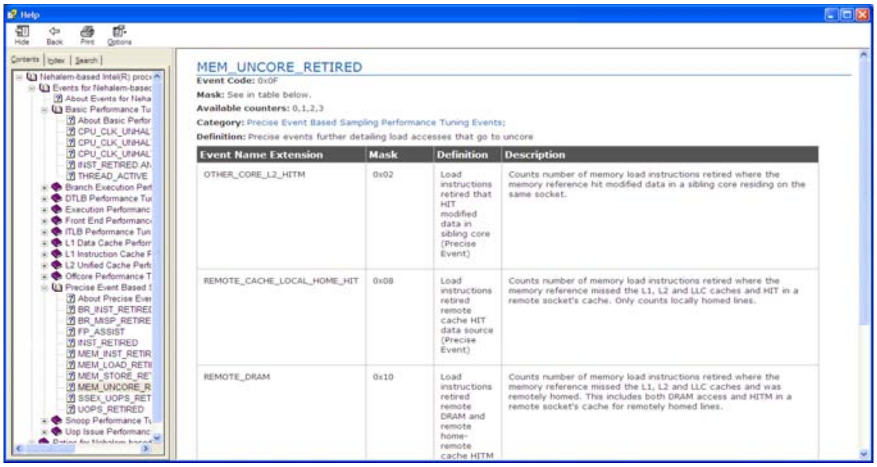
"miss" events are inclusive

– Sum over all data sources and their individual latencies

Intel® Xeon™ 5500 processor Precise events are exclusive

Per data source

Data Access Events Reveal NUMA Access Pattern



MEM_UNCORE_RETIRED

Event Code: 0x0F

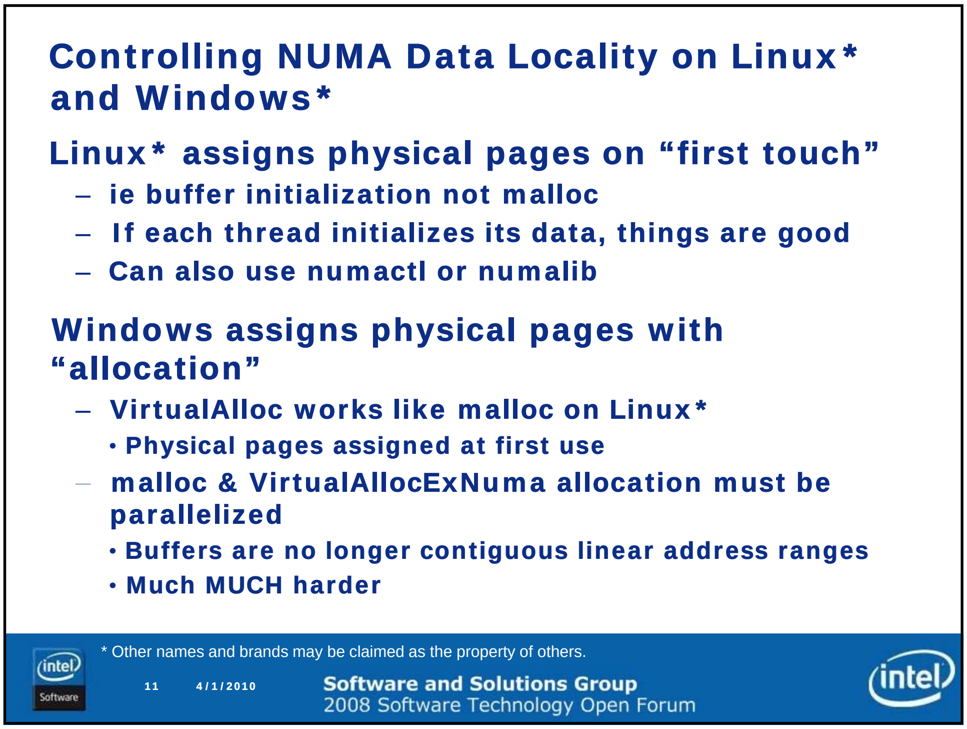
Mask: See in table below.

Available counters: 0,1,2,3

Category: Precise Event Based Sampling Performance Tuning Events;

Definition: Precise events further detailing load accesses that go to uncore

Event Name Extension	Mask	Definition	Description
OTHER_CORE_L2_HITM	0x02	Load instructions retired that HIT modified data in sibling core (Precise Event)	Counts number of memory load instructions retired where the memory reference hit modified data in a sibling core residing on the same socket.
REMOTE_CACHE_LOCAL_HOME_HIT	0x08	Load instructions retired remote cache HIT data source (Precise Event)	Counts number of memory load instructions retired where the memory reference missed the L1, L2 and LLC caches and HIT in a remote socket's cache. Only counts locally homed lines.
REMOTE_DRAM	0x10	Load instructions retired remote DRAM and remote home-remote cache HITM	Counts number of memory load instructions retired where the memory reference missed the L1, L2 and LLC caches and was remotely homed. This includes both DRAM access and HITM in a remote socket's cache for remotely homed lines.



Controlling NUMA Data Locality on Linux* and Windows*

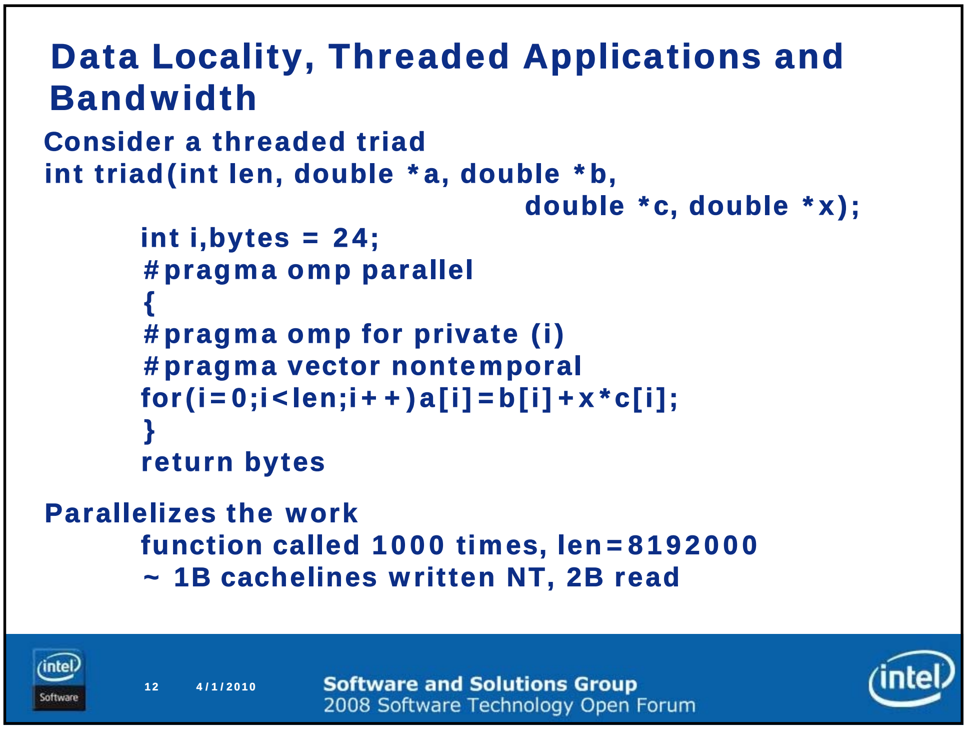
Linux* assigns physical pages on “first touch”

- ie buffer initialization not malloc
- If each thread initializes its data, things are good
- Can also use numactl or numalib

Windows assigns physical pages with “allocation”

- VirtualAlloc works like malloc on Linux*
  - Physical pages assigned at first use
- malloc & VirtualAllocExNuma allocation must be parallelized
  - Buffers are no longer contiguous linear address ranges
  - Much MUCH harder

* Other names and brands may be claimed as the property of others.



Data Locality, Threaded Applications and Bandwidth

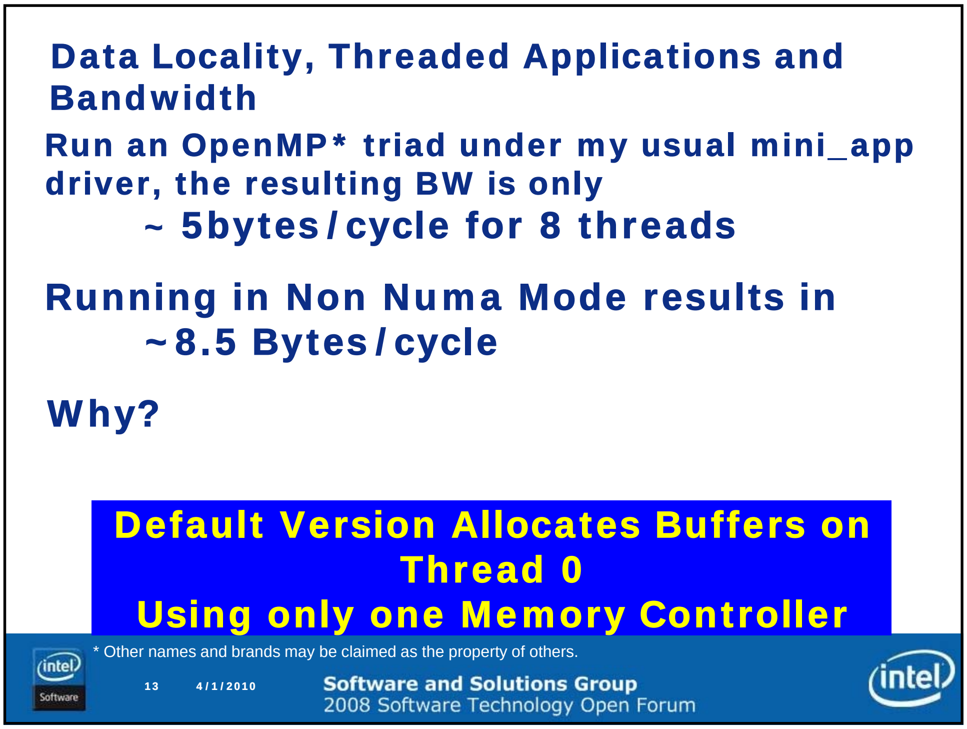
Consider a threaded triad

```
int triad(int len, double *a, double *b,
                                        double *c, double *x);
        int i,bytes = 24;
        #pragma omp parallel
        {
        #pragma omp for private (i)
        #pragma vector nontemporal
        for(i=0;i<len;i++)a[i]=b[i]+x*c[i];
        }
        return bytes
```

Parallelizes the work

function called 1000 times, len=8192000

~ 1B cachelines written NT, 2B read



Data Locality, Threaded Applications and Bandwidth

Run an OpenMP* triad under my usual mini_app driver, the resulting BW is only

~ 5bytes/cycle for 8 threads

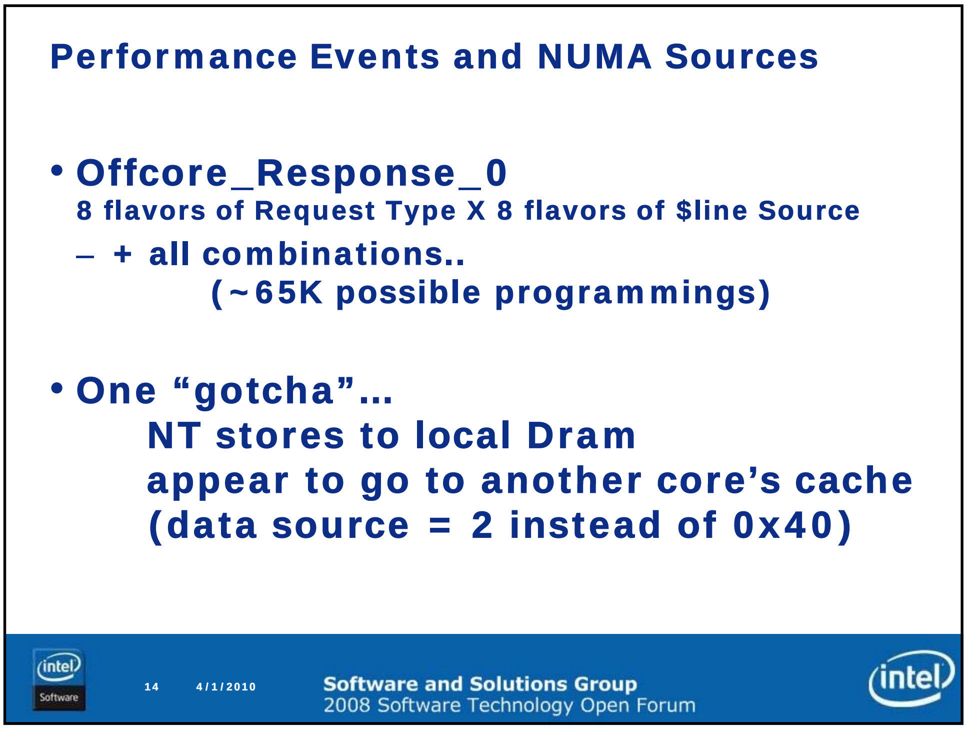
Running in Non Numa Mode results in

~8.5 Bytes/cycle

Why?

Default Version Allocates Buffers on Thread 0
Using only one Memory Controller

* Other names and brands may be claimed as the property of others.

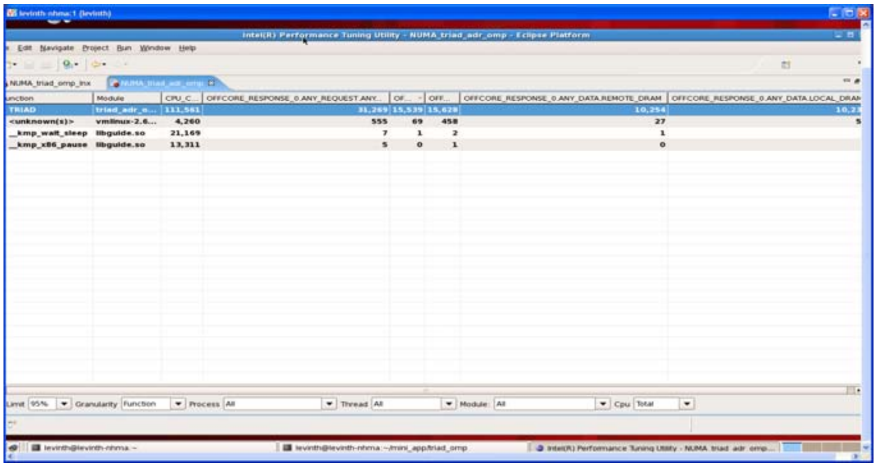


Performance Events and NUMA Sources

- Offcore_Response_0
  8 flavors of Request Type X 8 flavors of $line Source
  – + all combinations..
  (~65K possible programmings)

- One “gotcha”…
  NT stores to local Dram
  appear to go to another core’s cache
  (data source = 2 instead of 0x40)

PTU Display Shows Local and Remote Access for OpenMP Triad



levinth-nhma:1 (levinth)
Intel(R) Performance Tuning Utility - NUMA_triad_adr_omp - Eclipse Platform
Edit Navigate Project Run Window Help
NUMA_triad_omp_lnx
NUMA_triad_adr_omp

unction	Module	CPU_C...	OFFCORE_RESPONSE_0.ANY_REQUEST.ANY...	OF...	OFF...	OFFCORE_RESPONSE_0.ANY_DATA.REMOTE_DRAM	OFFCORE_RESPONSE_0.ANY_DATA.LOCAL_DRAM
TRIAD	triad_adr_o...	111,561	31,269	15,539	15,628	10,254	10,23
<unknown(s)>	vmlinux-2.6...	4,260	555	69	458	27	5
__kmp_wait_sleep	libguide.so	21,169	7	1	2	1	
__kmp_x86_pause	libguide.so	13,311	5	0	1	0	

Limit 95% Granularity Function Process All Thread All Module: All Cpu Total

levinth@levinth-nhma:~
levinth@levinth-nhma:~/mini_app/triad_omp
Intel(R) Performance Tuning Utility - NUMA triad adr omp...

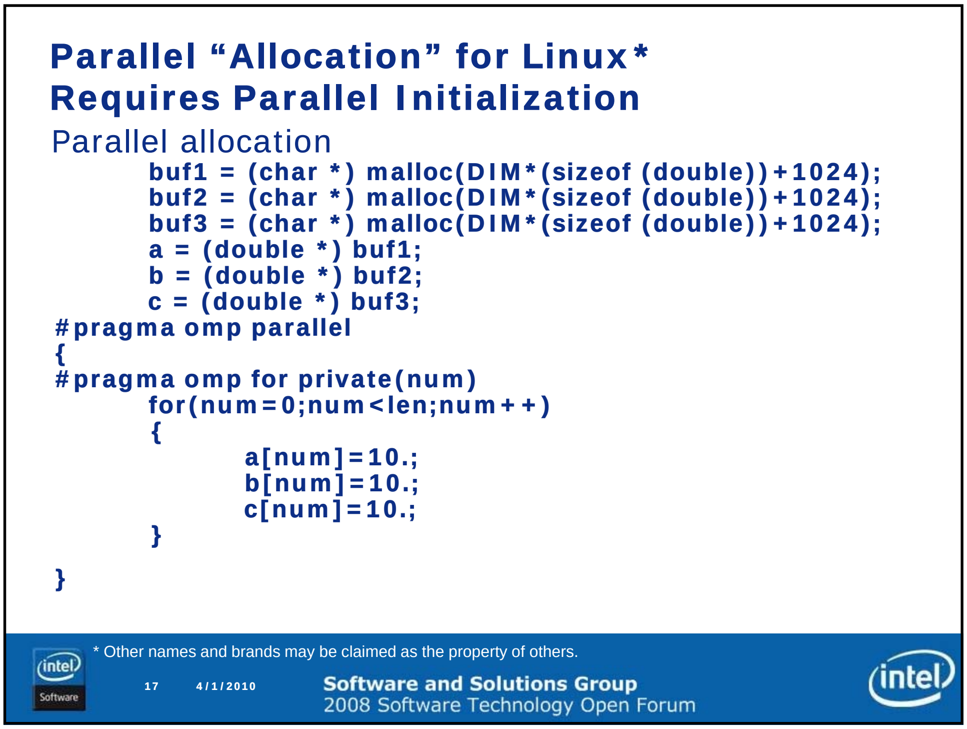
Need to Distribute "Allocation"
"Allocate" on First Touch

Original allocation

```
buf1 = (char *) malloc(DIM*(sizeof (double))+1024);
buf2 = (char *) malloc(DIM*(sizeof (double))+1024);
buf3 = (char *) malloc(DIM*(sizeof (double))+1024);
a = (double *) buf1;
b = (double *) buf2;
c = (double *) buf3;
for(num=0;num<len;num++)
{
        a[num]=10.;
        b[num]=10.;
        c[num]=10.;
}
```

Initialization must also be done in Parallel

* Other names and brands may be claimed as the property of others.

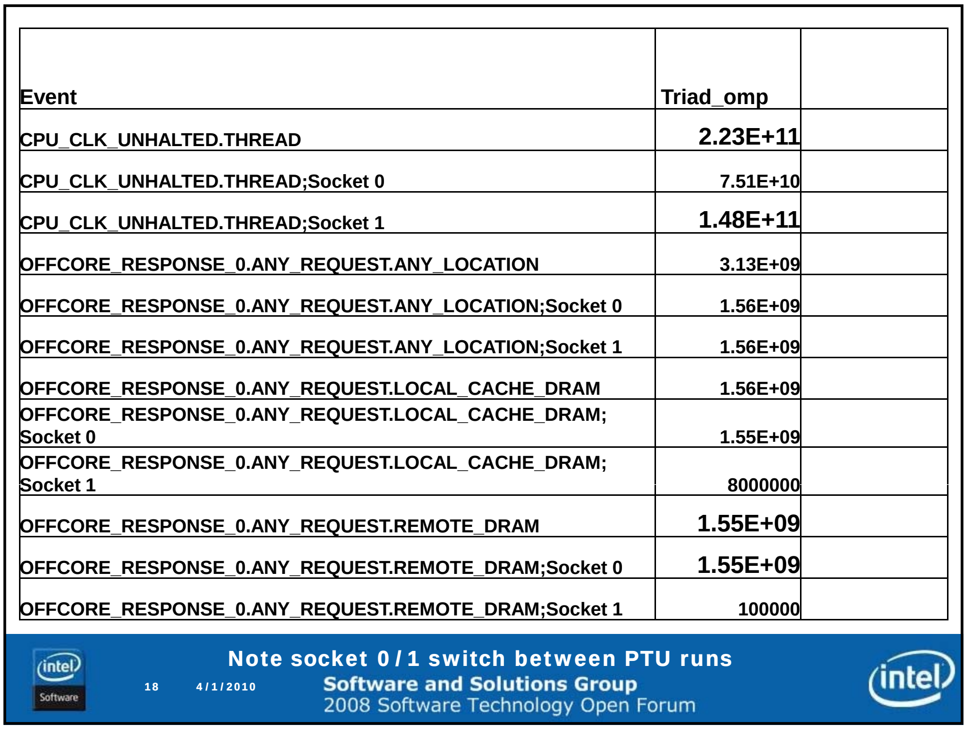


Parallel “Allocation” for Linux* Requires Parallel Initialization

Parallel allocation

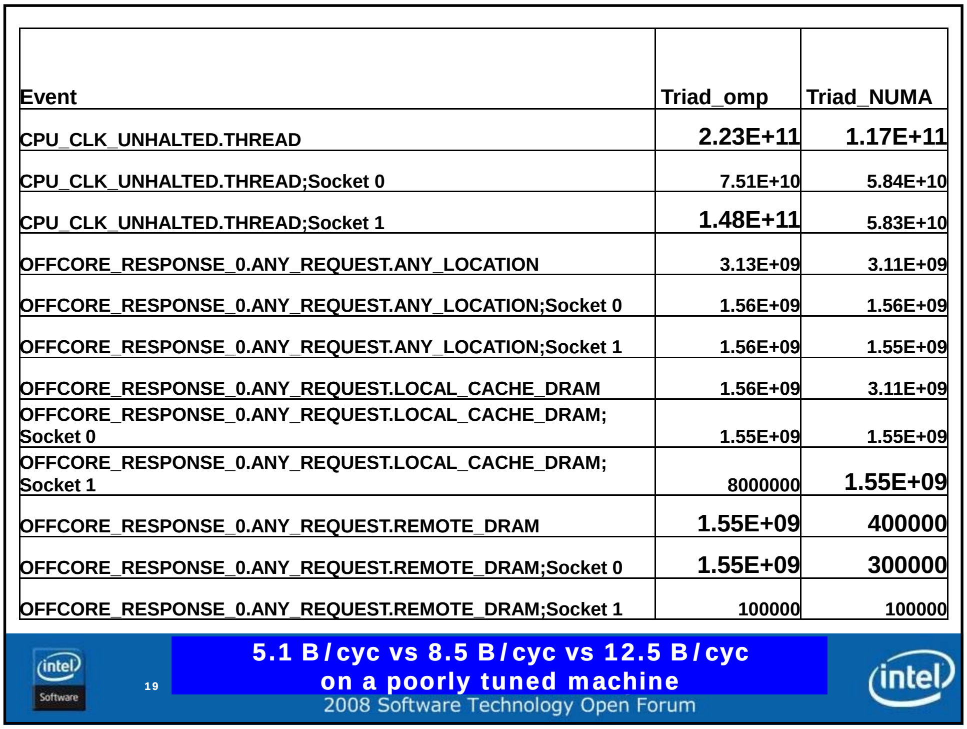
```
        buf1 = (char *) malloc(DIM*(sizeof (double))+1024);
        buf2 = (char *) malloc(DIM*(sizeof (double))+1024);
        buf3 = (char *) malloc(DIM*(sizeof (double))+1024);
        a = (double *) buf1;
        b = (double *) buf2;
        c = (double *) buf3;
#pragma omp parallel
{
#pragma omp for private(num)
        for(num=0;num<len;num++)
        {
                a[num]=10.;
                b[num]=10.;
                c[num]=10.;
        }

}
```

* Other names and brands may be claimed as the property of others.



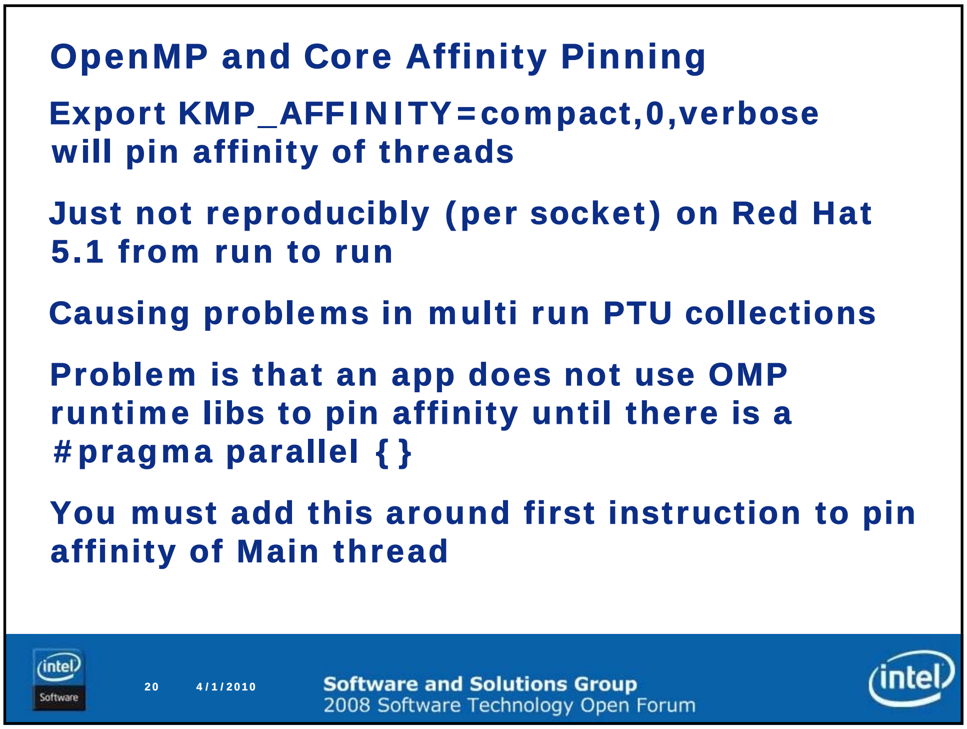
Event	Triad_omp	
CPU_CLK_UNHALTED.THREAD	2.23E+11	
CPU_CLK_UNHALTED.THREAD;Socket 0	7.51E+10	
CPU_CLK_UNHALTED.THREAD;Socket 1	1.48E+11	
OFFCORE_RESPONSE_0.ANY_REQUEST.ANY_LOCATION	3.13E+09	
OFFCORE_RESPONSE_0.ANY_REQUEST.ANY_LOCATION;Socket 0	1.56E+09	
OFFCORE_RESPONSE_0.ANY_REQUEST.ANY_LOCATION;Socket 1	1.56E+09	
OFFCORE_RESPONSE_0.ANY_REQUEST.LOCAL_CACHE_DRAM	1.56E+09	
OFFCORE_RESPONSE_0.ANY_REQUEST.LOCAL_CACHE_DRAM; Socket 0	1.55E+09	
OFFCORE_RESPONSE_0.ANY_REQUEST.LOCAL_CACHE_DRAM; Socket 1	8000000	
OFFCORE_RESPONSE_0.ANY_REQUEST.REMOTE_DRAM	1.55E+09	
OFFCORE_RESPONSE_0.ANY_REQUEST.REMOTE_DRAM;Socket 0	1.55E+09	
OFFCORE_RESPONSE_0.ANY_REQUEST.REMOTE_DRAM;Socket 1	100000	

Note socket 0/1 switch between PTU runs



Event	Triad_omp	Triad_NUMA
CPU_CLK_UNHALTED.THREAD	2.23E+11	1.17E+11
CPU_CLK_UNHALTED.THREAD;Socket 0	7.51E+10	5.84E+10
CPU_CLK_UNHALTED.THREAD;Socket 1	1.48E+11	5.83E+10
OFFCORE_RESPONSE_0.ANY_REQUEST.ANY_LOCATION	3.13E+09	3.11E+09
OFFCORE_RESPONSE_0.ANY_REQUEST.ANY_LOCATION;Socket 0	1.56E+09	1.56E+09
OFFCORE_RESPONSE_0.ANY_REQUEST.ANY_LOCATION;Socket 1	1.56E+09	1.55E+09
OFFCORE_RESPONSE_0.ANY_REQUEST.LOCAL_CACHE_DRAM	1.56E+09	3.11E+09
OFFCORE_RESPONSE_0.ANY_REQUEST.LOCAL_CACHE_DRAM; Socket 0	1.55E+09	1.55E+09
OFFCORE_RESPONSE_0.ANY_REQUEST.LOCAL_CACHE_DRAM; Socket 1	8000000	1.55E+09
OFFCORE_RESPONSE_0.ANY_REQUEST.REMOTE_DRAM	1.55E+09	400000
OFFCORE_RESPONSE_0.ANY_REQUEST.REMOTE_DRAM;Socket 0	1.55E+09	300000
OFFCORE_RESPONSE_0.ANY_REQUEST.REMOTE_DRAM;Socket 1	100000	100000

5.1 B/cyc vs 8.5 B/cyc vs 12.5 B/cyc on a poorly tuned machine



OpenMP and Core Affinity Pinning

Export KMP_AFFINITY=compact,0,verbose will pin affinity of threads

Just not reproducibly (per socket) on Red Hat 5.1 from run to run

Causing problems in multi run PTU collections

Problem is that an app does not use OMP runtime libs to pin affinity until there is a #pragma parallel {}

You must add this around first instruction to pin affinity of Main thread

Multi-thread Scaling and NUMA

When measuring scaling between 4 and 8 threads (assuming no SMT) the affinity of the 4 threads matters

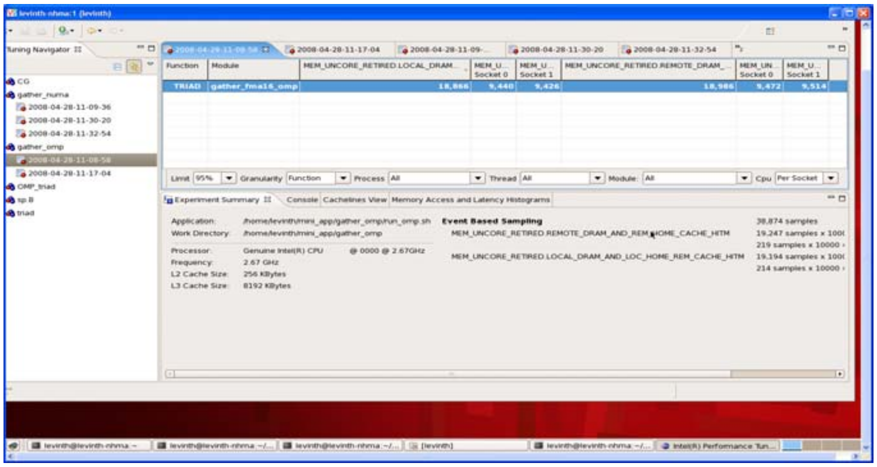
4 threads all on one socket has the same LLC cache size/core as 8 threads

BUT

2 threads/socket has closer to the same memory BW as the 8 thread run

Thus 4->8 scaling will always have a non scaling contribution due to one of these 2 effects

Per Socket Display + Data Source events Show NUMA /Cross Socket Traffic



Function	Module	MEM_UNCORE_RETIRED.LOCAL_DRAM...	MEM_U... Socket 0	MEM_U... Socket 1	MEM_UNCORE_RETIRED.REMOTE_DRAM_...	MEM_UN... Socket 0	MEM_U... Socket 1
TRIAD	gather_fma16_omp	18,866	9,440	9,426	18,986	9,472	9,514

Limit 95% Granularity Function Process All Thread All Module: All Cpu Per Socket

Application: /home/levinth/mini_app/gather_omp/run_omp.sh
Work Directory: /home/levinth/mini_app/gather_omp

Processor: Genuine Intel(R) CPU @ 0000 @ 2.67GHz
Frequency: 2.67 GHz
L2 Cache Size: 256 KBytes
L3 Cache Size: 8192 KBytes

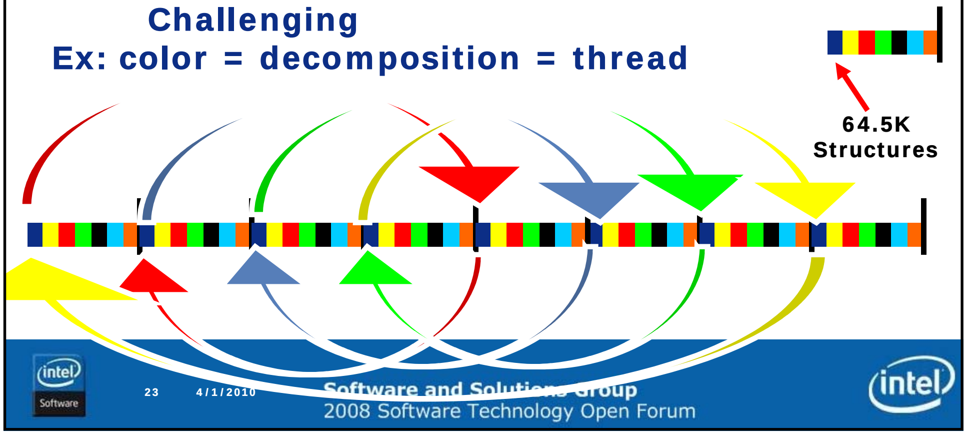
Event Based Sampling 38,874 samples
MEM_UNCORE_RETIRED.REMOTE_DRAM_AND_REM_HOME_CACHE_HITM 19,247 samples x 100(
219 samples x 10000 ›
MEM_UNCORE_RETIRED.LOCAL_DRAM_AND_LOC_HOME_REM_CACHE_HITM 19,194 samples x 100(
214 samples x 10000 ›

Indirect Addressing, Locality and Latency (Diff Eq on Non Uniform Grid, Oil Res)

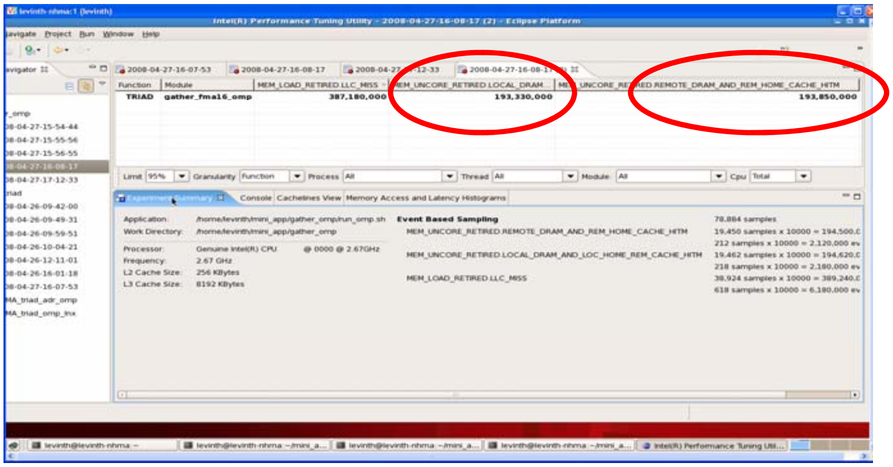
Multi-dimensional array access can cause large address gaps in data decomposition.

This can make mapping NUMA home node->pages->data decomposition ranges Challenging

Ex: color = decomposition = thread



Default Initialization Breaks Array into 8 Contiguous Pieces→ 50% Non Local Access

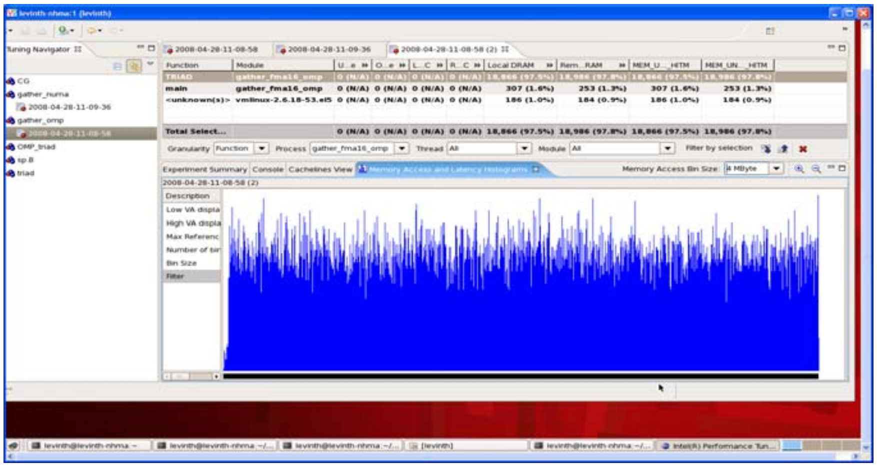


Function	Module	MEM_LOAD_RETIRED.LLC_MISS	MEM_UNCORE_RETIRED.LOCAL_DRAM...	MEM_UNCORE_RETIRED.REMOTE_DRAM_AND_REM_HOME_CACHE_HITM
TRIAD	gather_fma16_omp	387,180,000	193,330,000	193,850,000

Limit 95% Granularity Function Process All Thread All Module All Cpu Total

```
Application:      /home/levinth/mini_app/gather_omp/run_omp.sh   Event Based Sampling                                            78,884 samples
Work Directory:   /home/levinth/mini_app/gather_omp                MEM_UNCORE_RETIRED.REMOTE_DRAM_AND_REM_HOME_CACHE_HITM       19,450 samples x 10000 = 194,500,0
                                                                                                                                  212 samples x 10000 = 2,120,000 ev
Processor:        Genuine Intel(R) CPU      @ 0000 @ 2.67GHz       MEM_UNCORE_RETIRED.LOCAL_DRAM_AND_LOC_HOME_REM_CACHE_HITM    19,462 samples x 10000 = 194,620,0
Frequency:        2.67 GHz                                                                                                        218 samples x 10000 = 2,180,000 ev
L2 Cache Size:    256 KBytes                                       MEM_LOAD_RETIRED.LLC_MISS                                    38,924 samples x 10000 = 389,240,0
L3 Cache Size:    8192 KBytes                                                                                                     618 samples x 10000 = 6,180,000 ev
```

Address Histogram for all Dram Accesses

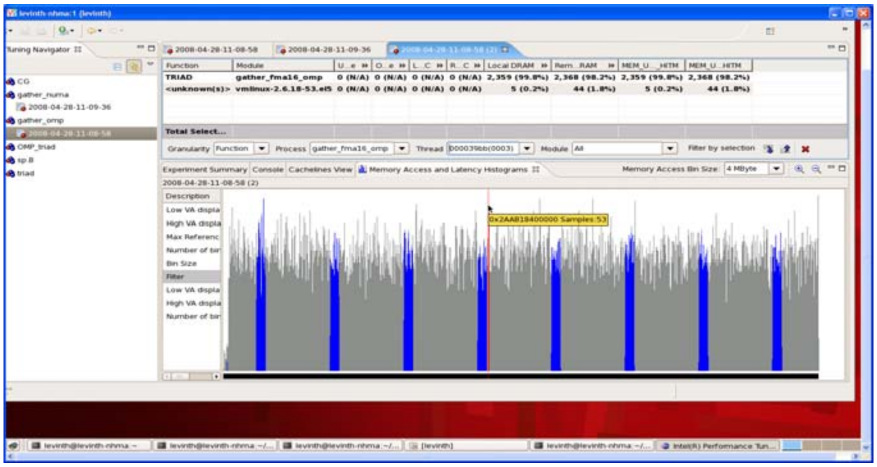


Function	Module	U...e	O...e	L...C	R...C	Local DRAM	Rem...RAM	MEM_U..._HITM	MEM_UN..._HITM
TRIAD	gather_fma16_omp	0 (N/A)	0 (N/A)	0 (N/A)	0 (N/A)	18,866 (97.5%)	18,986 (97.8%)	18,866 (97.5%)	18,986 (97.8%)
main	gather_fma16_omp	0 (N/A)	0 (N/A)	0 (N/A)	0 (N/A)	307 (1.6%)	253 (1.3%)	307 (1.6%)	253 (1.3%)
<unknown(s)>	vmlinux-2.6.18-53.el5	0 (N/A)	0 (N/A)	0 (N/A)	0 (N/A)	186 (1.0%)	184 (0.9%)	186 (1.0%)	184 (0.9%)
Total Select...		0 (N/A)	0 (N/A)	0 (N/A)	0 (N/A)	18,866 (97.5%)	18,986 (97.8%)	18,866 (97.5%)	18,986 (97.8%)

Granularity: Function   Process: gather_fma16_omp   Thread: All   Module: All   Filter by selection

Memory Access Bin Size: 4 MByte

Filtering to a Single Thread Displays the Data Decomposition

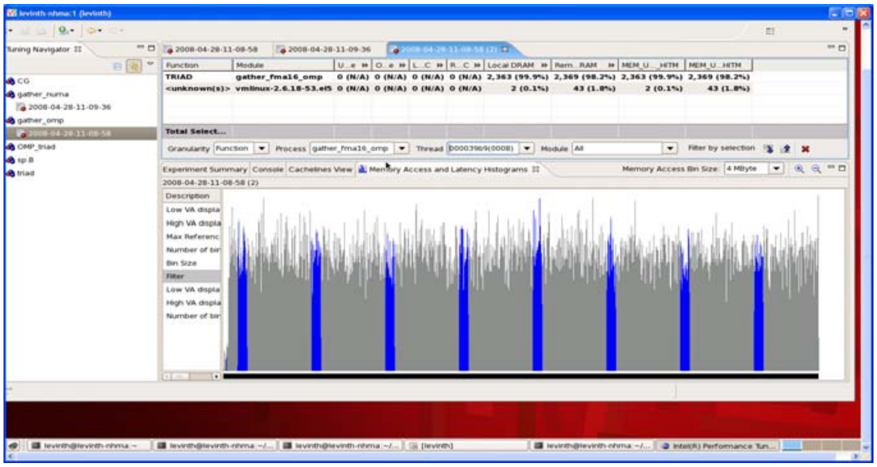


Function	Module	U...e	O...e	L...C	R...C	Local DRAM	Rem...RAM	MEM_U..._HITM	MEM_U..._HITM
TRIAD	gather_fma16_omp	0 (N/A)	0 (N/A)	0 (N/A)	0 (N/A)	2,359 (99.8%)	2,368 (98.2%)	2,359 (99.8%)	2,368 (98.2%)
<unknown(s)>	vmlinux-2.6.18-53.el5	0 (N/A)	0 (N/A)	0 (N/A)	0 (N/A)	5 (0.2%)	44 (1.8%)	5 (0.2%)	44 (1.8%)
Total Select...									

Granularity: Function   Process: gather_fma16_omp   Thread: 000039bb(0003)   Module: All   Filter by selection

Memory Access Bin Size: 4 MByte

A Different Thread



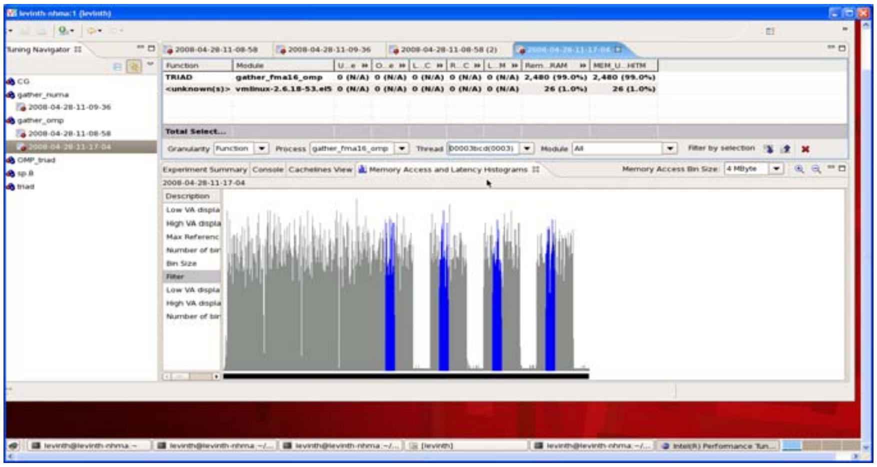
Function	Module	U...e	O...e	L...C	R...C	Local DRAM	Rem...RAM	MEM_U..._HITM	MEM_U...HITM
TRIAD	gather_fma16_omp	0 (N/A)	0 (N/A)	0 (N/A)	0 (N/A)	2,363 (99.9%)	2,369 (98.2%)	2,363 (99.9%)	2,369 (98.2%)
<unknown(s)>	vmlinux-2.6.18-53.el5	0 (N/A)	0 (N/A)	0 (N/A)	0 (N/A)	2 (0.1%)	43 (1.8%)	2 (0.1%)	43 (1.8%)

Granularity: Function Process: gather_fma16_omp Thread: 0000398(0008) Module: All

Memory Access Bin Size: 4 MByte

2008-04-28-11-08-58 (2)

Using Only Precise Remote Dram Event
Only Half the entries shown
Gaps due to lack of events are suppressed



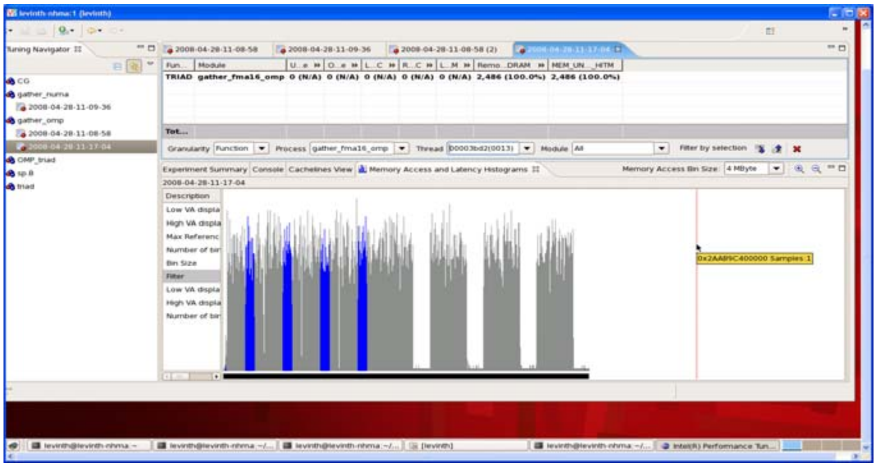
Function	Module	U...e	O...e	L...C	R...C	L...M	Rem...RAM	MEM_U...HITM
TRIAD	gather_fma16_omp	0 (N/A)	0 (N/A)	0 (N/A)	0 (N/A)	0 (N/A)	2,480 (99.0%)	2,480 (99.0%)
<unknown(s)>	vmlinux-2.6.18-53.el5	0 (N/A)	0 (N/A)	0 (N/A)	0 (N/A)	0 (N/A)	26 (1.0%)	26 (1.0%)

Granularity: Function Process: gather_fma16_omp Thread: 00003bcd(0003) Module: All

Memory Access Bin Size: 4 MByte

2008-04-28-11-17-04

Using Only Precise Remote Dram Event
Only Half the entries shown
Gaps due to lack of events are suppressed



Fun...	Module	U...e	O...e	L...C	R...C	L...M	Remo...DRAM	MEM_UN..._HITM
TRIAD	gather_fma16_omp	0 (N/A)	0 (N/A)	0 (N/A)	0 (N/A)	0 (N/A)	2,486 (100.0%)	2,486 (100.0%)

Granularity: Function Process: gather_fma16_omp Thread: 00003bd2(0013) Module: All

Memory Access Bin Size: 4 MByte

Change Initialization to Follow Work Access Pattern

Thread initialization with same access sequence as work

Expect ~33% improvement

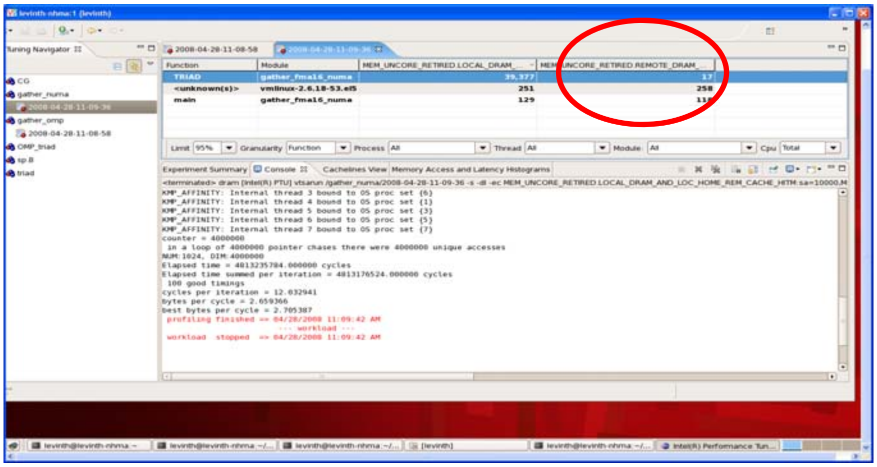
- 1/2 of accesses get lower latency by 2

Simple OMP ran in 14.3 cycles/cell

NUMA initialized version ran in 11.2 cycles/cell

Every access has serious DTLB issues, which don’t change with the improved NUMA layout

Sampling View for Correctly Initialized Array has no Remote Access



Function	Module	MEM_UNCORE_RETIRED.LOCAL_DRAM_...	MEM_UNCORE_RETIRED.REMOTE_DRAM_...
TRIAD	gather_fma16_numa	39,377	17
<unknown(s)>	vmlinux-2.6.18-53.el5	251	258
main	gather_fma16_numa	129	11[illegible]

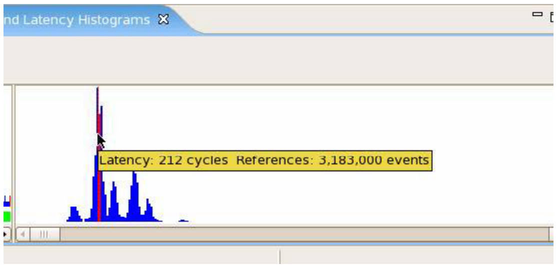
Limit: 95% Granularity: Function Process: All Thread: All Module: All Cpu: Total

```
<terminated> dram [Intel(R) PTU] vtsarun /gather_numa/2008-04-28-11-09-36 -s -dl -ec MEM_UNCORE_RETIRED.LOCAL_DRAM_AND_LOC_HOME_REM_CACHE_HITM:sa=10000,M
KMP_AFFINITY: Internal thread 3 bound to OS proc set {6}
KMP_AFFINITY: Internal thread 4 bound to OS proc set {1}
KMP_AFFINITY: Internal thread 5 bound to OS proc set {3}
KMP_AFFINITY: Internal thread 6 bound to OS proc set {5}
KMP_AFFINITY: Internal thread 7 bound to OS proc set {7}
counter = 4000000
 in a loop of 4000000 pointer chases there were 4000000 unique accesses
NUM:1024, DIM:4000000
Elapsed time = 4813235784.000000 cycles
Elapsed time summed per iteration = 4813176524.000000 cycles
 100 good timings
cycles per iteration = 12.032941
bytes per cycle = 2.659366
best bytes per cycle = 2.705387
 profiling finished => 04/28/2008 11:09:42 AM
                         --- workload ---
 workload  stopped  => 04/28/2008 11:09:42 AM
```

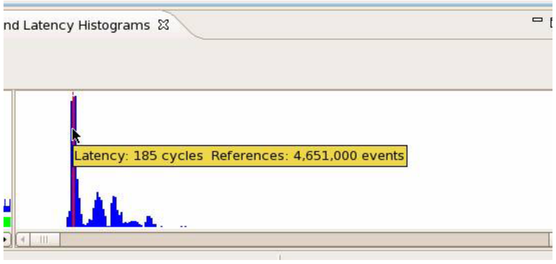
Page Allocation Order Matters

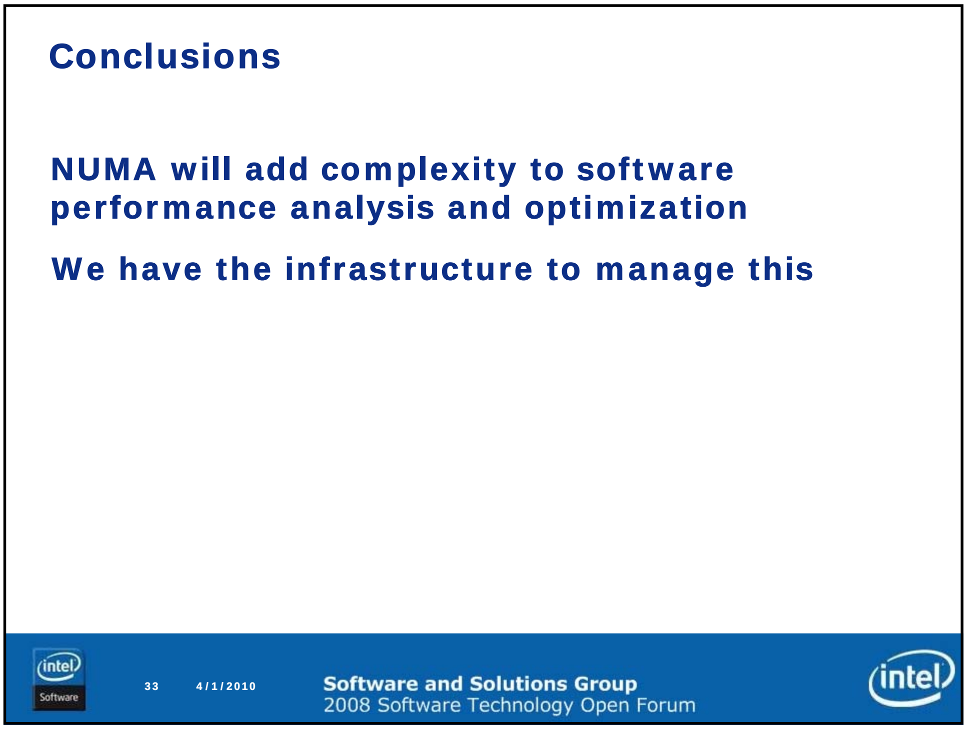
Serially initialized/allocated

Accessed with complex pattern avg Lat =230



Initialized/Allocated and Accessed with complex pattern avg Lat = 209

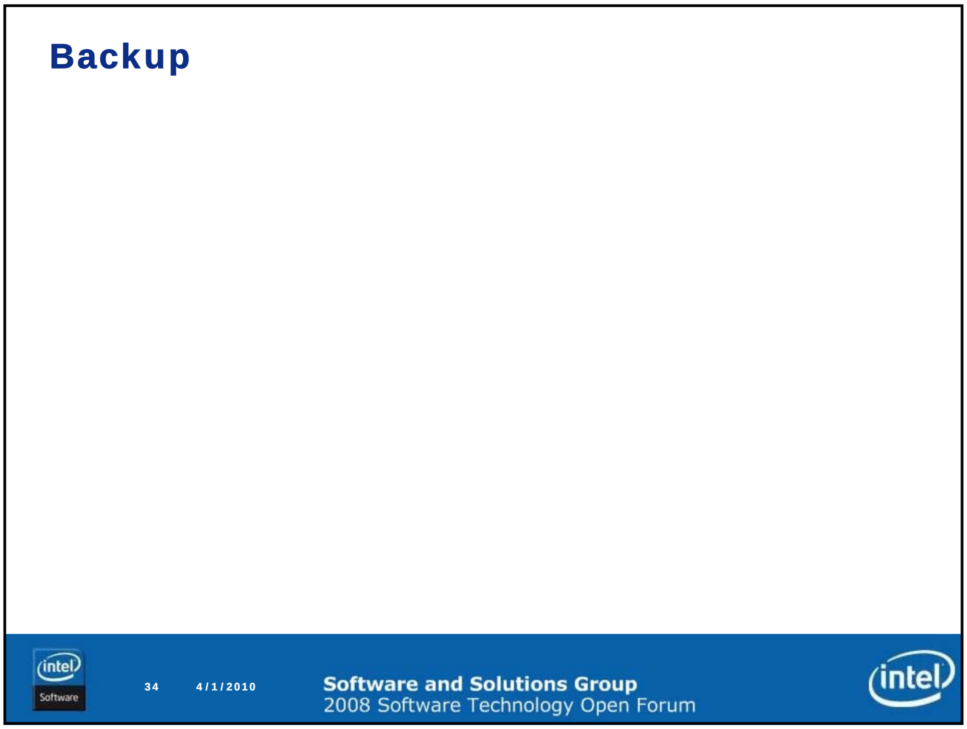




Conclusions

NUMA will add complexity to software performance analysis and optimization

We have the infrastructure to manage this



Backup