CERN Openlab

CERN openlab

NUMA experience on large many-core servers from AMD/Intel

Julien Leduc CERN openlab julien.leduc at cern.ch

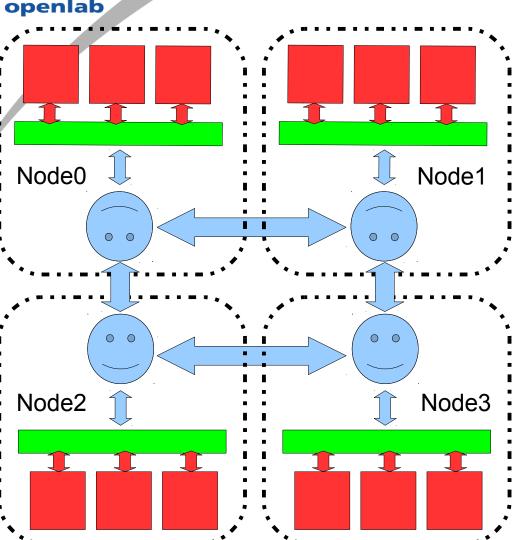


Current servers architectures

Enterprise servers divide in 3 categories:

- SMP (Symmetric Multi Processing)
 - Common architecture multiple processors connected symmetrically on the memory system
- MPP (Massive Parallel Processing)
 - Non sharing architecture dividing the system into several nodes that can access local resources often connected with proprietary interconnect
- NUMA (Non Uniform Memory Access)
 - The full system divides into multiple nodes which can access both local and remote memory.

NUMA architecture



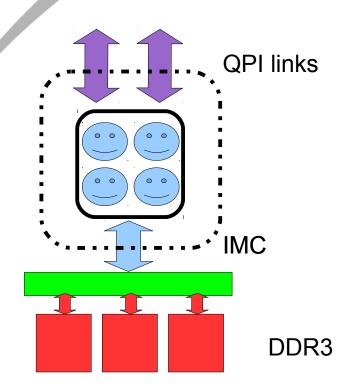
NUMA architecture divides in multiple nodes with access to local and remote memory, at a cost for remote memory.

Why NUMA now?



- Increasing performance now means more and more cores
- Both CPUs and memory don't see a boost in frequency anymore
- NUMA nicely solves these issues, offering a scalable design for future many-cores systems



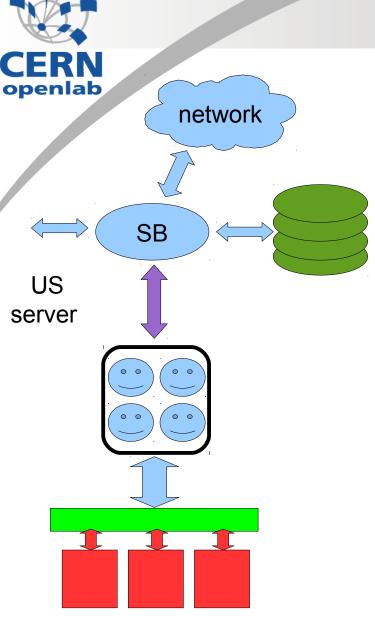


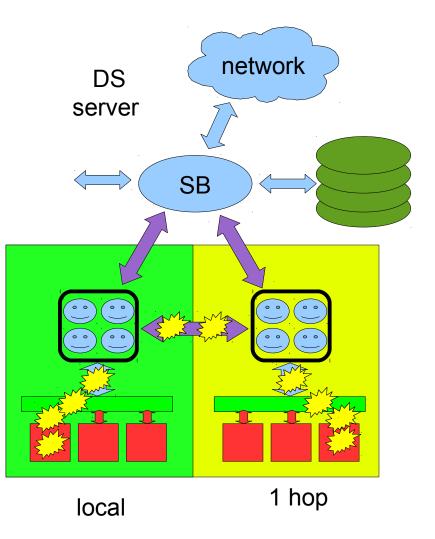
Dual Socket capable NHM

Nehalem microarchitecture

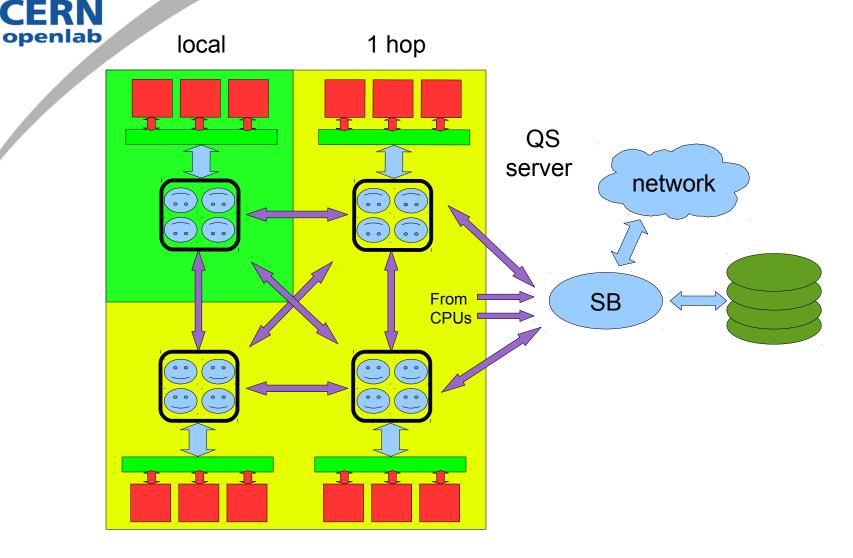
- Nehalem microarchitecture is equipped with an Integrated Memory Controller, and some QuickPath Interconnect links (1 for workstations, 2 for DS servers, 4 for MS servers)
- Microarchitecture allows scalable design for servers

Nehalem microarchitecture Single and Dual Sockets designs





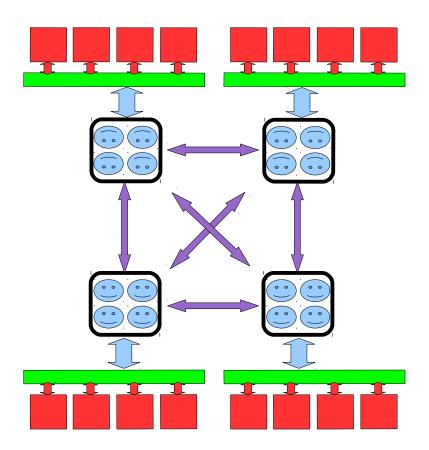
Nehalem microarchitecture Quad Sockets design





Magny-Cours Quad Socket design

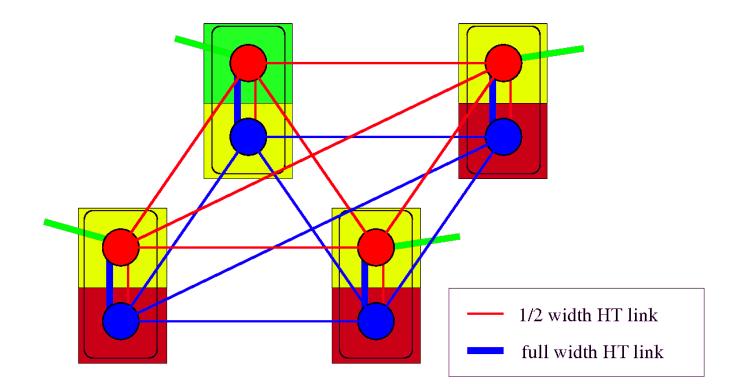
 Each 12-core Magny-Cours is connected to its 3 other neighbors





Magny-cours Quad Socket design

 But each Magny-Cours is composed of a pair of 6-core Istanbul CPUs



NUMA considerations



NUMA factor:

- (remote latency)/(local latency)
- for Westmere 140ns/90ns ~ 1.5
- Linux decomposes a Numa server into nodes:
 - A node is a set of CPUs and its associated memory given by ACPI tables
 - DP system
 QP system

available: 2 nodes (0-1) node 0 size: 12279 MB node 1 size: 12288 MB node distances: node 0 1 0: 10 20

1: 20 10

available: 4 nodes (0-3) node 0 size: 32209 MB node 1 size: 32320 MB node 2 size: 32320 MB node 3 size: 32320 MB node distances: node 0 1 2 3

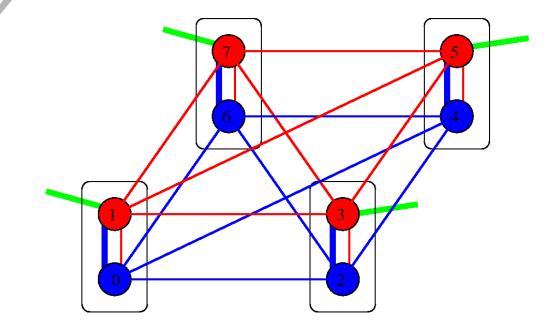
- 0: 10 21 21 21
- 1: 21 10 21 21 2: 21 21 10 21
- 3: 21 21 21 10

NUMA considerations

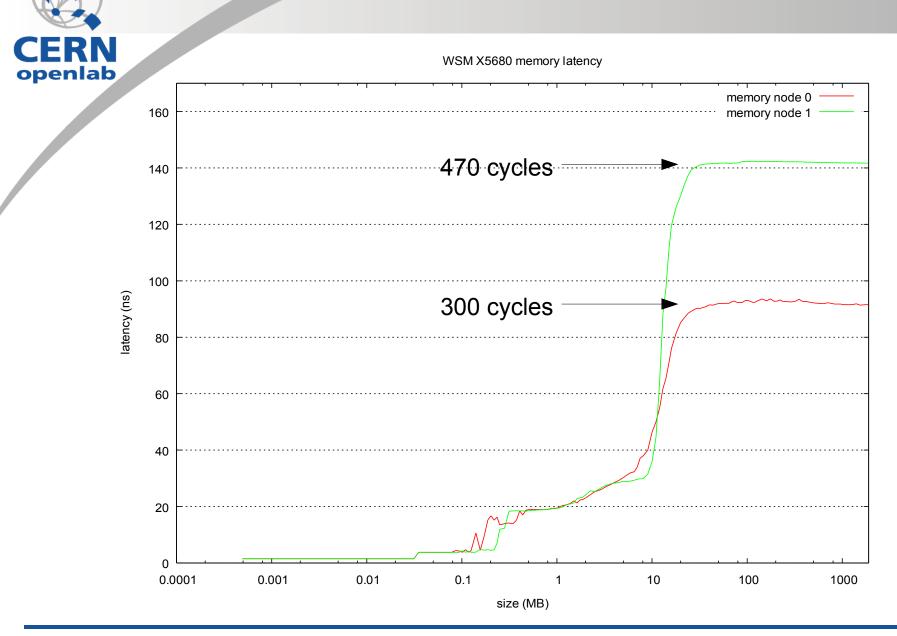
QP Magny-Cours

CERN

openlab



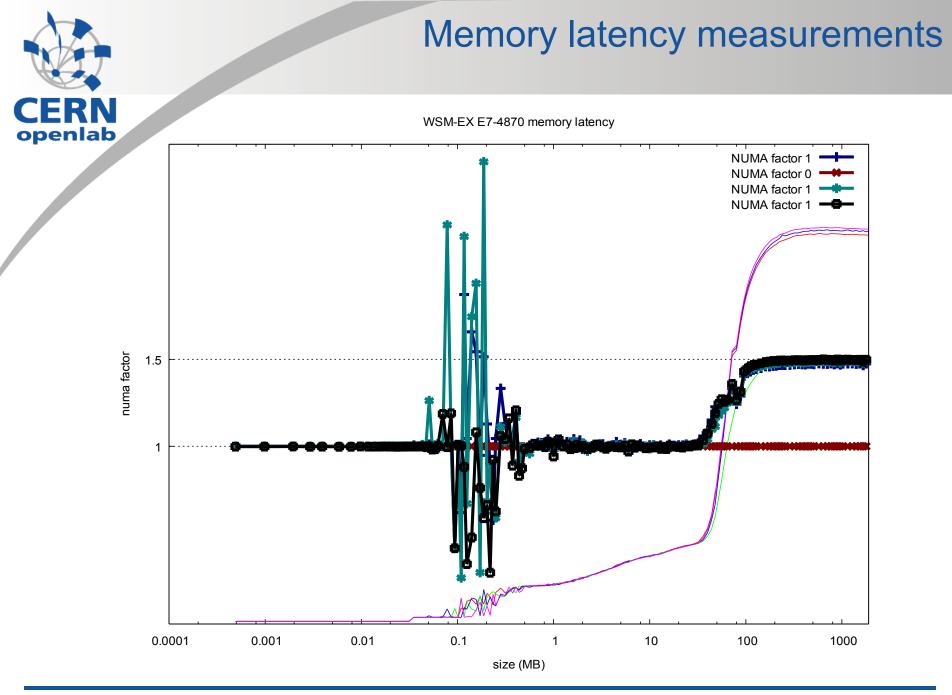
Memory latency measurements



Memory latency measurements WSM X5680 memory latency openlab NUMA factor 1.5 1 latency (ns) 0.0001 0.001 0.01 0.1 1 10 100 1000 size (MB)

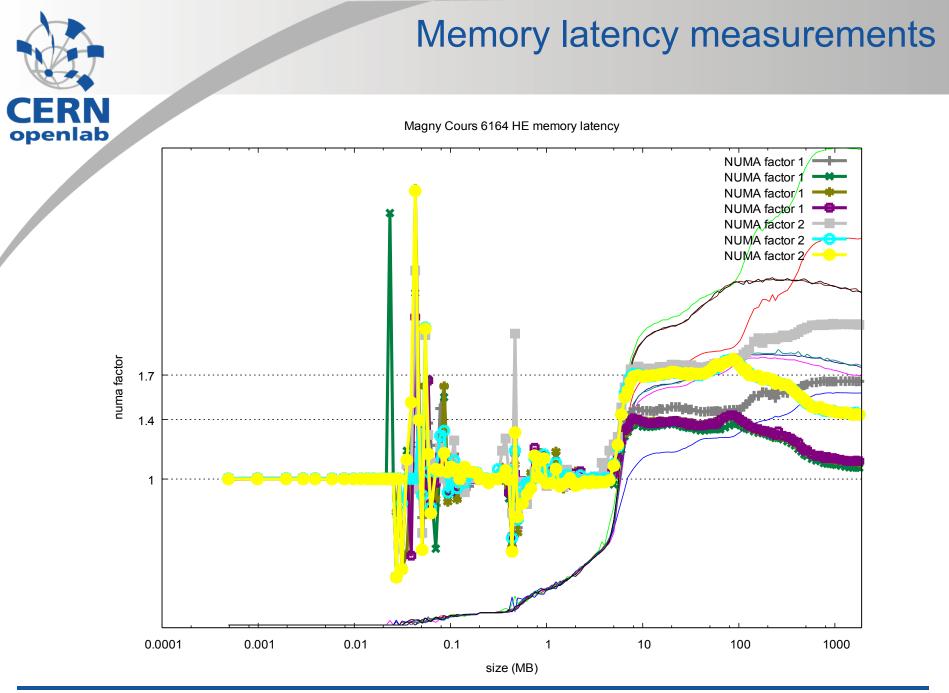
Memory latency measurements

WSM-EX E7-4870 memory latency openlab memory node 0 memory node 1 250 memory node 2 memory node 3 530 cycles 200 360 cycles latency (ns) 150 100 50 0 0.0001 0.001 0.01 0.1 10 100 1000 1 size (MB)



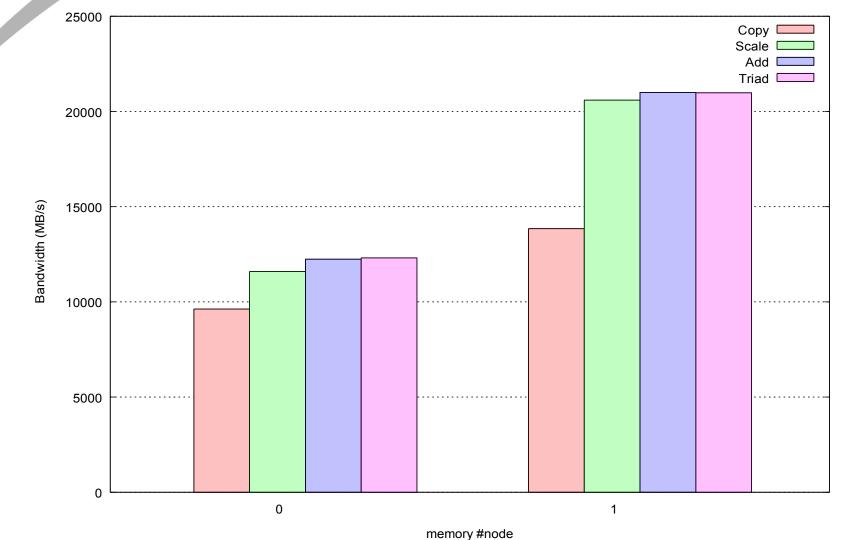
Memory latency measurements

Magny Cours 6164 HE memory latency openlab 350 memory node 0 memory node 1 memory node 2 memory node 3 300 memory node 4 memory node 5 memory node 6 memory node 7 250 380 cycles 200 latency (ns) 300 cycles 150 220 cycles 100 50 0 0.0001 0.001 0.01 0.1 10 100 1000 1 size (MB)



WSM X5680 memory bandwidth stream OMP 6 threads on cpu node 1

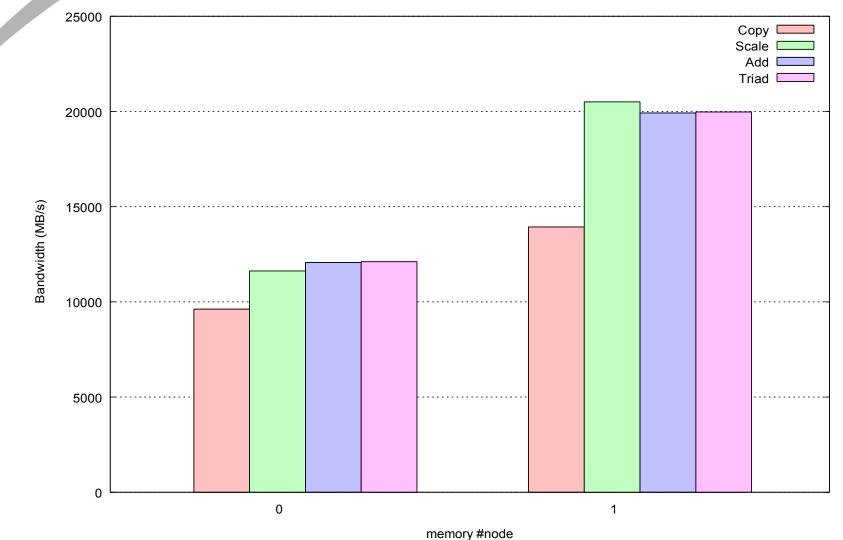
openlab



Second International Workshop for Future Challenges in Tracking and Trigger Concepts

WSM X5680 memory bandwidth HT-on stream OMP 12 threads on cpu node 1

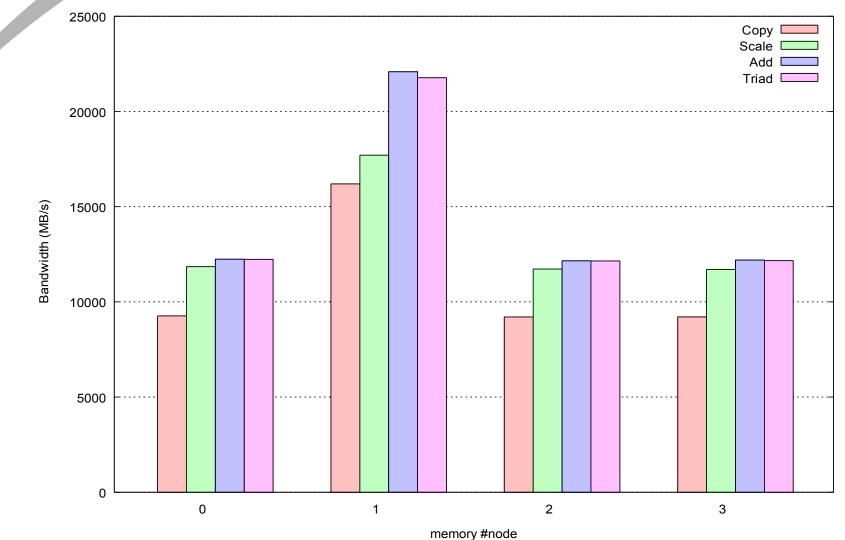
openlab



Second International Workshop for Future Challenges in Tracking and Trigger Concepts

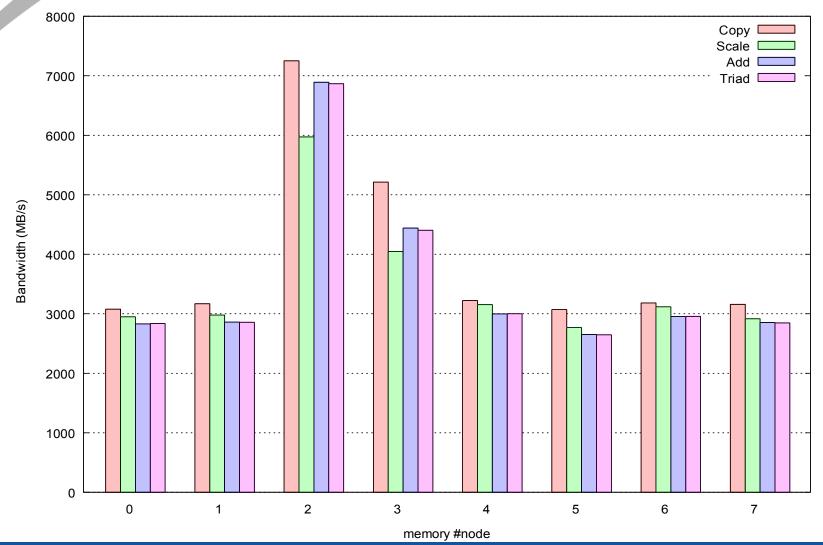
WSM-EX E7-4870 memory bandwidth stream OMP 10 threads on cpu node 1

openlab



Magny Cours 6164 HE memory bandwidth stream OMP 6 threads on cpu node 2

openlab





Managing OpenMP on NUMA systems

- OpenMP eases parallel application development
 - Perfect for SMP systems from previous generations
 - Flat memory model
 - What do NUMA systems change?

stream



 Stream is a simple bandwidth benchmark, a lot of implementations are available

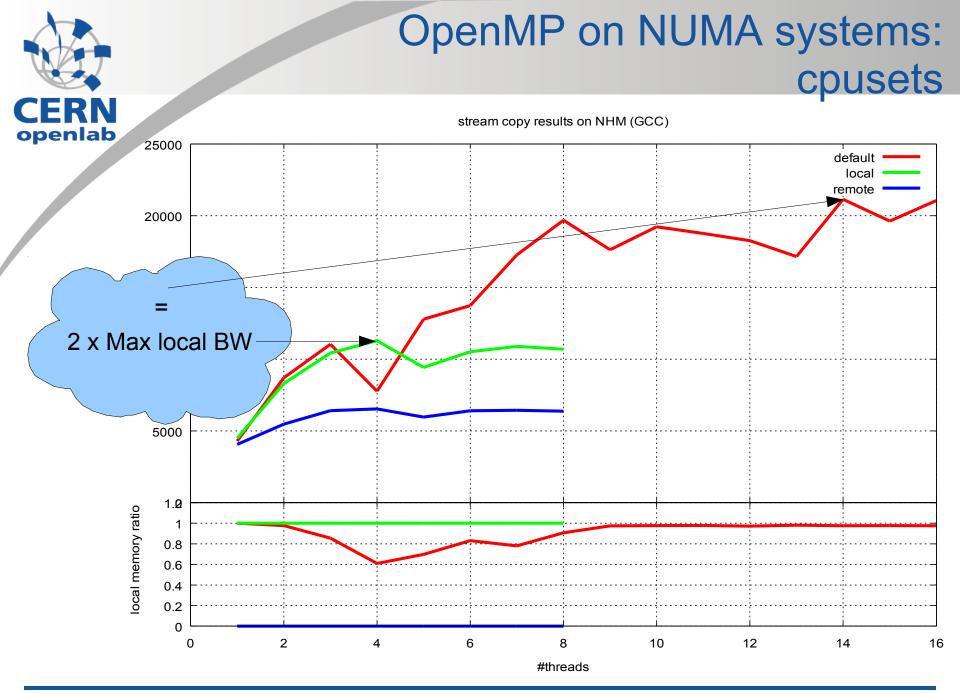
- Single threaded
- OpenMP
- MPI
- Customs

3 large arrays of doubles defined (N=20000k)

- Using 2000000*8*3=457MB of data
- 4 operations performed and timed on those arrays:
 - Copy: c[j] = a[j]
 - Scale: b[j] = scalar * c[j]
 - Add: c[j] = a[j] + b[j]
 - Triad: a[j] = b[j] + scalar * c[j]



- First compile stream omp with GCC
- Using CPUSET, stream is constrained to run on cores of one numa node and use:
 - Local memory: allocating memory on the same numa node
 - Remote memory: allocating memory on the other numa node
- How does stream behaves when running on the system without any constraint





OpenMP on NUMA systems

- Using ICC, environment variable KMP_AFFINITY allows to control omp threads placement
 - Verbose: allows to extract the scheduling information
 - granularity=core: control placement at the core level
 - Type:
 - **Compact**: assigns the OpenMP thread <n>+1 to a free thread context as close as possible to the thread context where the <n> OpenMP thread was placed
 - **Scatter**: distributes the threads as evenly as possible across the entire system (opposite of compact)
- Compare execution and scheduling using KMP_AFFINITY:
 - Unset
 - KMP_AFFINITY=verbose,granularity=core,compact
 - KMP_AFFINITY=verbose,granularity=core,scatter

OpenMP on NUMA systems: KMP_AFFINITY

stream copy results on NHM (ICC) openlab default 35000 compact scatter 30000 25000 MB/s 20000 15000 10000 5000 1.0 local memory ratio 1 0.8 0.6 0.4 0.2 0 2 6 10 12 14 8 16 0 4 #threads

OpenMP on NUMA systems X5670 stream copy results on WSM (ICC) openlab default compact scatter MB/s

Second International Workshop for Future Challenges in Tracking and Trigger Concepts

#threads

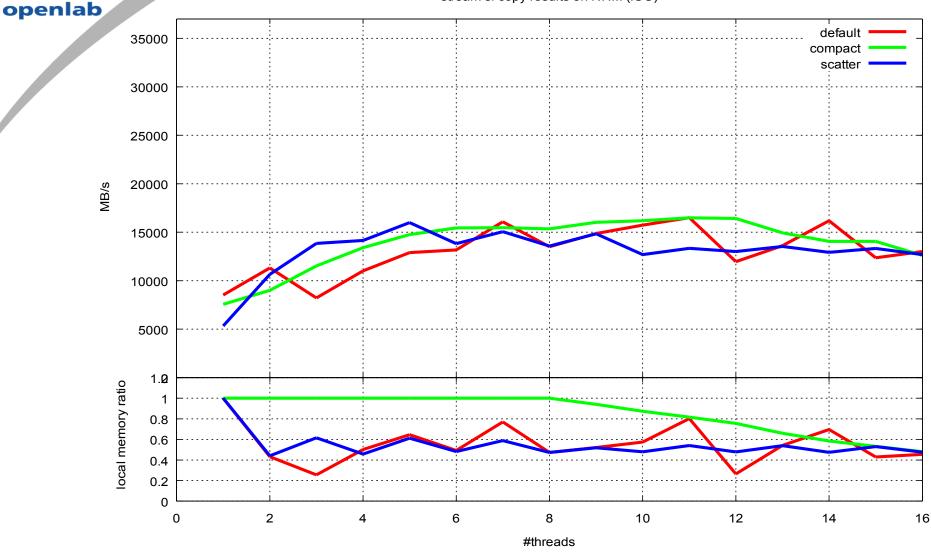


OpenMP on NUMA systems

- A hidden "detail":
 - During all the previous measurements, memory was initialized in parallel
 - What happens if memory is initialized by the master thread?

OpenMP on NUMA systems serial initialization

stream si copy results on NHM (ICC)



OpenMP on NUMA systems



- Parallel memory initialization is primordial because Linux and most operating systems use memory in place: where it was first touched
 - Memory migration patches exist but are not in the kernel
- Affinity can be interesting:
 - Using a numa DP system as 2 SMP systems in one server (compact, cpuset, numactl)
 - Scatter, can help
 - Explicit scheduling is possible

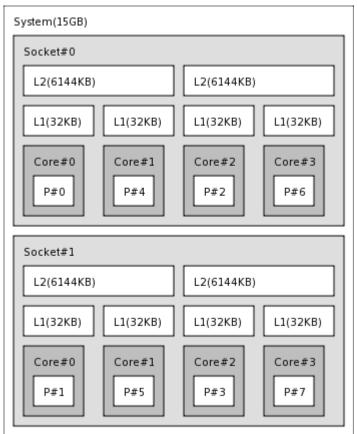
Need to extract the topology of the system



Extracting system topology

Hwloc

- Displays the topology (Istopo)
- Offers some bindings to cpusets



System(23GB)			
Node#0(11GB)			
Socket#0			
L3(8192KB)			
L2(256KB)	L2(256KB)	L2(256KB)	L2(256KB)
L1(32KB)	L1(32KB)	L1(32KB)	L1(32KB)
Core#0	Core#1	Core#2	Core#3
P#0 P#8	P#2 P#10	P#4 P#12	P#6 P#14
Node#1(12GB)			
Node#1(12GB)			
Node#1(12GB)	L2(256KB)	L2(256KB)	L2(256KB)
Node#1(12GB) Socket#1 L3(8192KB)		L2(256KB) L1(32KB)	L2(256KB) L1(32KB)
Node#1(12GB) Socket#1 L3(8192KB) L2(256KB)	L2(256KB)		
Node#1(12GB) Socket#1 L3(8192KB) L2(256KB) L1(32KB)	L2(256KB) L1(32KB)	L1(32KB)	L1(32KB)

Extracting system topology



• Westmere:

 6 cores, 2 QPI links per socket, Integrated Memory Controller, 32nm

System(11GB)					
Node#0(6029MB)					
Socket#0					
L3(12MB)					
L2(256KB)	L2(256KB)	L2(256KB)	L2(256KB)	L2(256KB)	L2(256KB)
L1(32KB)	L1(32KB)	L1(32KB)	L1(32KB)	L1(32KB)	L1(32KB)
Core#0 P#0 P#12	Core#1 P#1 P#13	Core#2 P#2 P#14	Core#8	Core#9 P#4 P#16	Core#10 P#5 P#17
Node#1(6060MB)					
Socket#1					
L3(12MB)					
L2(256KB)	L2(256KB)	L2(256KB)	L2(256KB)	L2(256KB)	L2(256KB)
L1(32KB)	L1(32KB)	L1(32KB)	L1(32KB)	L1(32KB)	L1(32KB)
Core#0 P#6 P#18	Core#1	Core#2	Core#8	Core#9	Core#10



WSM-EX

- 40 cores
- 4 sockets

lachine (1266B)	
NUMANode P#0 (31GB)	
Socket P#0	
L3 (30MB)	
L2 (256KB) L2 (256KB) <thl2 (256kb)<="" th=""> L2 (256KB) L2 (256K</thl2>	L2 (256KB) L2 (256KB)
L1 (32KB) L1 (32KB) <thl1 (32kb)<="" th=""> <thl1 (32kb)<="" th=""> <thl< td=""><td>L1 (32KB) L1 (32KB)</td></thl<></thl1></thl1>	L1 (32KB) L1 (32KB)
Core P#0 Core P#16 Core P#8 Core P#24 Core P#1 Core P#17 Core P#9 Core P#25	Core P#2 Core P#18
PU P#0 PU P#4 PU P#8 PU P#12 PU P#16 PU P#20 PU P#24 PU P#28 PU P#37 PU P#36 PU P#40 PU P#44 PU P#48 PU P#57 PU P#56	PU P#64 PU P#68
PU P#32 PU P#36 PU P#40 PU P#44 PU P#48 PU P#52 PU P#56 PU P#60	
NUMANode P#2 (32GB)	
Socket P#2	
L2 (256KB)	L2 (256KB) L2 (256KB)
L1 (32KB) L1 (32	L1 (32KB) L1 (32KB)
Core P#0 Core P#16 Core P#8 Core P#24 Core P#1 Core P#17 Core P#9 Core P#25	Core P#2 Core P#18
PU P#1 PU P#5 PU P#9 PU P#13 PU P#17 PU P#21 PU P#25 PU P#29 PU P#33 PU P#37 PU P#41 PU P#45 PU P#49 PU P#53 PU P#57 PU P#61	PU P#65 PU P#69 PU P#73 PU P#77
NUMANode P#1 (32GB)	
Socket P#1	
L2 (256KB)	L2 (256KB) L2 (256KB)
L1 (32KB) L1 (32	L1 (32KB)
Core P#0 Core P#16 Core P#8 Core P#24 Core P#1 Core P#17 Core P#9 Core P#25 PU P#2 PU P#6 PU P#10 PU P#14 PU P#18 PU P#22 PU P#26 PU P#30	Core P#2 Core P#18 PU P#66 PU P#70
PU P#34 PU P#38 PU P#42 PU P#46 PU P#50 PU P#54 PU P#58 PU P#62	PU P#74 PU P#78
NUMANode P#3 (32GB)	
Socket P#3	
L3 (30MB)	
L2 (256KB)	L2 (256KB) L2 (256KB)
L1 (32KB) L1 (32KB) L1 (32KB) L1 (32KB) L1 (32KB) L1 (32KB)	L1 (32KB) L1 (32KB)
Core P#0 Core P#16 Core P#24 Core P#1 Core P#17 Core P#9 Core P#25	Core P#2 Core P#18
PU P#3 PU P#7 PU P#11 PU P#15 PU P#19 PU P#23 PU P#27 PU P#31	PU P#67 PU P#71
PU P#35 PU P#39 PU P#43 PU P#47 PU P#51 PU P#55 PU P#59 PU P#63	PU P#75 PU P#79

Extract the topology



Magny-Cours

- 48 cores
- 4 sockets

NUMANode P#0) (12GB)				
_3 (5118KB)					
_2 (512KB)	L2 (512KB)	L2 (512KB)	L2 (512KB)	L2 (512KB)	L2 (512KB)
_1 (64KB)	L1 (64KB)	L1 (64KB)	L1 (64KB)	L1 (64KB)	L1 (64KB)
Core P#0	Core P#1	Core P#2	Core P#3	Core P#4	Core P#5
PU P#0	PU P#1	PU P#2	PU P#3	PU P#4	PU P#5
]
NUMANode P#1	(12GB)				
.3 (5118KB)					
_2 (512KB)	L2 (512KB)	L2 (512KB)	L2 (512KB)	L2 (512KB)	L2 (512KB)
_1 (64KB)	L1 (64KB)	L1 (64KB)	L1 (64KB)	L1 (64KB)	L1 (64KB)
Core P#0	Core P#1	Core P#2	Core P#3	Core P#4	Core P#5
PU P#6	PU P#7	PU P#8	PU P#9	PU P#10	PU P#11
cet P#3 (24GB)					
NUMANode P#4	↓(12GB)				
_3 (5118KB)					
1	L2 (512KB)	L2 (512KB)	L2 (512KB)	L2 (512KB)	L2 (512KB)
_3 (5118КВ) _2 (512КВ) _1 (64КВ)	L2 (512KB) L1 (64KB)	L2 (512KB)	L2 (512KB)	L2 (512KB)	L2 (512KB) L1 (64KB)
_2 (512KB)					
_2 (512КВ) _1 (64КВ)	L1 (64KB)	L1 (64KB)	L1 (64KB)	L1 (64KB)	L1 (64KB)
.2 (512KB) .1 (64KB) Core P#0	L1 (64KB)	L1 (64KB)	L1 (64KB)	L1 (64KB)	L1 (64KB)
2 (512KB) 1 (64KB) Core P#0 PU P#24	L1 (64KB) Core P#1 PU P#25	L1 (64KB)	L1 (64KB)	L1 (64KB)	L1 (64KB)
.2 (512KB) .1 (64KB) Core P#0	L1 (64KB) Core P#1 PU P#25	L1 (64KB)	L1 (64KB)	L1 (64KB)	L1 (64KB)
2 (512KB) 1 (64KB) 2 ore P#0 PU P#24 NUMANode P#5 3 (5118KB)	L1 (64KB) Core P#1 PU P#25	L1 (64KB)	L1 (64KB)	L1 (64KB)	L1 (64KB)
.2 (512KB) .1 (64KB) Core P#0 PU P#24	L1 (64KB) Core P#1 PU P#25	L1 (64KB) Core P#2 PU P#26	L1 (64KB) Core P#3 PU P#27	L1 (64KB) Core P#4 PU P#28	L1 (64KB) Core P#5 PU P#29
2 (512KB) .1 (64KB) Core P#0 PU P#24 .3 (5118KB) .2 (512KB)	L1 (64KB) Core P#1 PU P#25 ; (12GB) L2 (512KB)	L1 (64KB) Core P#2 PU P#26	L1 (64KB) Core P#3 PU P#27 L2 (512KB)	L1 (64KB) Core P#4 PU P#28	L1 (64KB) Core P#5 PU P#29

Extract the topology

NUMANode P#2	(12GB)				
L3 (5118KB)					
L2 (512KB)	L2 (512KB)	L2 (512KB)	L2 (512KB)	L2 (512KB)	L2 (512K
L1 (64KB)	L1 (64KB)	L1 (64KB)	L1 (64KB)	L1 (64KB)	L1 (64KE
Core P#0 PU P#12	Core P#1 PU P#13	Core P#2 PU P#14	Core P#3 PU P#15	Core P#4 PU P#16	Core P#
NUMANode P#3	(12GB)				
L3 (5118KB)					
L2 (512KB)	L2 (512KB)	L2 (512KB)	L2 (512KB)	L2 (512KB)	L2 (512
L1 (64KB)	L1 (64KB)	L1 (64KB)	L1 (64KB)	L1 (64KB)	L1 (64K
Core P#0	Core P#1	Core P#2	Core P#3	Core P#4	Core P#
PU P#18	PU P#19	PU P#20	PU P#21	PU P#22	PU P#
		PU P#20	PU P#21	PU P#22	PU P#
ocket P#4 (23GB)		PU P#20	PU P#21	PU P#22	PU P#
ocket P#4 (23GB)		PU P#20	PU P#21	PU P#22	
ocket P#4 (23GB) NUMANode P#6 L3 (5118KB)	(12G8)				L2 (512
ocket P#4 (23GB) NUMANode P#6 L3 (5118KB) L2 (512KB)	(12GB)	L2 (512KB)	L2 (512KB)	L2 (512KB)	L2 (512 L1 (64K
ocket P#4 (23G8) NUMANode P#6 L3 (5118KB) L2 (512KB) L1 (64KB) Core P#0	(12GB) L2 (512KB) L1 (64KB) Core P#1 PU P#37	L2 (512KB) L1 (64KB) Core P#2	L2 (512KB) L1 (64KB) Core P#3	L2 (512KB) L1 (64KB) Core P#4	L2 (512 L1 (64K
ocket P#4 (23G8) NUMANode P#6 L3 (5118KB) L2 (512KB) L1 (64KB) Core P#0 PU P#36	(12GB) L2 (512KB) L1 (64KB) Core P#1 PU P#37	L2 (512KB) L1 (64KB) Core P#2	L2 (512KB) L1 (64KB) Core P#3	L2 (512KB) L1 (64KB) Core P#4	L2 (512 L1 (64K
iocket P#4 (23GB) NUMANode P#6 L3 (5118KB) L2 (512KB) L1 (64KB) Core P#0 PU P#36 NUMANode P#7	(12GB) L2 (512KB) L1 (64KB) Core P#1 PU P#37	L2 (512KB) L1 (64KB) Core P#2	L2 (512KB) L1 (64KB) Core P#3	L2 (512KB) L1 (64KB) Core P#4	L2 (512 L1 (64K Core P4 PU P#
iocket P#4 (23G8) NUMANode P#6 L3 (5118KB) L2 (512KB) L1 (64KB) Core P#0 PU P#36 NUMANode P#7 L3 (5118KB)	(12GB) L2 (512KB) L1 (64KB) Core P#1 PU P#37 (11GB)	L2 (512KB) L1 (64KB) Core P#2 PU P#38	L2 (512KB) L1 (64KB) Core P#3 PU P#39	L2 (512KB) L1 (64KB) Core P#4 PU P#40	Р Р Р Р Р Р Р Р Р Р Р Р Р Р Р Р Р Р Р



Questions?