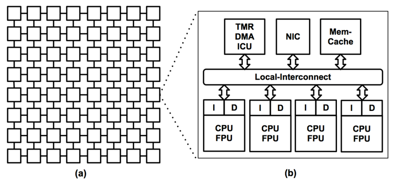
On the impact of many-cores small caches on the scalability of shared-memory highly multi-threaded single-applications

Ghassan Almaless - Franck Wajsbürt

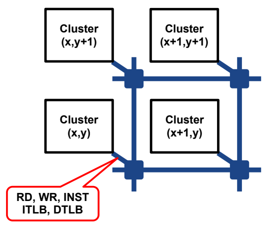
LIP6 - UPMC Sorbonne Universités – 4, place Jussieu – Paris, France – firstname.lastname@lip6.fr

TSAR: Tera-Scale Architecture



- Defined by LIP6 & BULL (European MEDEA+)
- Clustered Architecture with 2D mesh NoC
- cc-NUMA scalable up to +1024 cores
- Each core has its own MMU and TLB
- Shared distributed physical address space

Hardware Event Counters



Goal

- Investigate if there is bad allocation or placement of physical pages

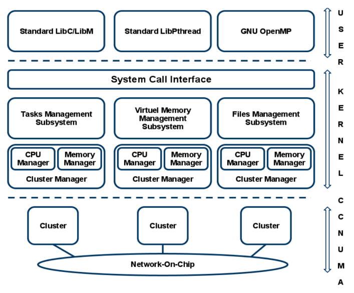
Per-cluster incoming L2-cache requests

RD	L1 data miss
INST	L1 instruction miss
WR	data write (write-through)
ITLB	TLB instruction miss
DTLB	TLB data miss
LL	Linked Load
SC	Store Conditional

Classification of incoming requests

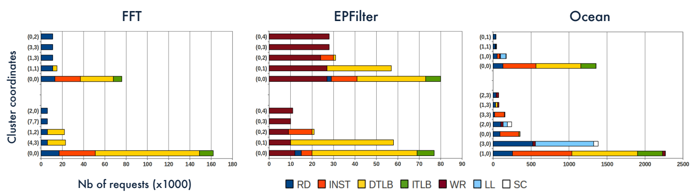
- **Payload**: RD, WR, LL, SC
- **Overhead**: INST, ITLB, DTLB

ALMOS: Advanced Locality Management OS



- New research Unix-like OS
- cc-NUMA dedicated
- Enforce the localilty of memory accesses
- Legacy unmodified applications are supported
- Similar threading model and implementation as Linux

Scalability Limitation Analysis

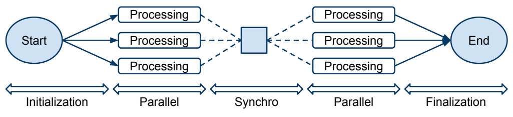


Histogram of remote cache-related requests received by a cluster for the last two scalability points

- Per cluster reduction of incoming payload request
- Increase of overhead request received mainly by cluster (0,0)

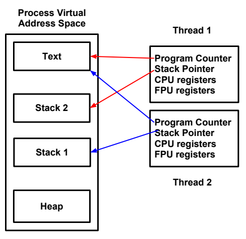
Overhead treatment serialization is the bottleneck

Evaluated Workloads



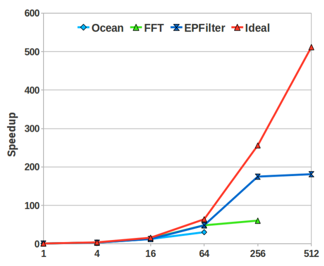
- Unmodified HPC Applications
  - FFT & Ocean (SPLASH-2)
  - EPFilter (Philips medical-image filtering)
- Highly multi-threaded single applications based on PThreads
- Small execution time suitable for full-system accurate emulation
- 3 inter-threads communication scheme (NoC & caches stress)
  - All-to-all writes (EPFilter)
  - All-to-all reads (FFT)
  - Neighborhood reads & writes (Ocean)

Scalability Drawback of Current Notion of Threads



- The overhead-request cannot be eliminated
- DTLB and ITLB are related to the notion of virtual address space
- Process virtual address space is defined by a single page tables
- All threads share their process address space
- All threads refer to the same page tables
- **The observed bottleneck is inherently related to this current notion of threads**

Experimental Results



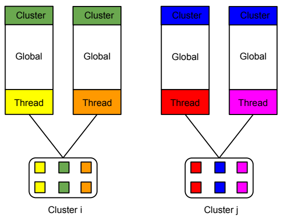
TSAR configuration

- Core type: MIPS32
- L1-I & L1-D *16kB, 4-ways, 64B cache-line*
- TLB-I & TLB-D *16 entries, 4-ways*
- Memory-Cache (L2) *256kB, 16-ways*
- Mesh of 8x16 *128 clusters, 512 cores*

Problem sizes

- Ocean (contiguous): grid of 514x514
- FFT: 262144 (M=18) complex doubles
- EPFilter: 1024x1024 pix (CF of 201x35 pix)

Solution in Principle



Goal: breaking the bottleneck by

- Replicating the page tables
- Replicating the instructions

Each thread has its own virtual address space with 3 regions

- Thread-private (e.g. stack)
- Cluster-shared (e.g. instructions)
- Global-shared (e.g. heap)

Kernel-level solution

- Transparent to userland
- Conform to POSIX Threads standard

APSYS 2012
Asia-Pacific Workshop on Systems