From Classical to Runtime Aware Architectures

Prof. Mateo Valero
BSC Director

Workshop Syec 25-26 April
Technological Achievements

Transistor (Bell Labs, 1947)
- DEC PDP-1 (1957)
- IBM 7090 (1960)

Integrated circuit (1958)
- IBM System 360 (1965)
- DEC PDP-8 (1965)

Microprocessor (1971)
- Intel 4004
Birth of the Revolution – The Intel 4004

Introduced November 15, 1971
108KHz, 50 KIPs, 2300 10μ transistors
Sunway TaihuLight

- SW26010 processor (Chinese design, ISA, & fab)
- 1.45 GHz
- Node = 260 Cores (1 socket)
- 4 – core groups
- 32 GB memory
- 40,960 nodes in the system
- 10,649,600 cores total
- 1.31 PB of primary memory (DDR3).
- 125.4 Pflop/s theoretical peak
- 93 Pflop/s HPL, 74% peak
- 15.3 Mwatts water cooled
- 3 of the 6 finalists for Gordon Bell Award@SC16
# Top 500 Supercomputers - November 2016

<table>
<thead>
<tr>
<th>Rank</th>
<th>Name</th>
<th>Site</th>
<th>Computer</th>
<th>Total Cores</th>
<th>Rmax</th>
<th>Rpeak</th>
<th>Power</th>
<th>Mflops/W</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sunway TaihuLight</td>
<td>National Supercomputing Center in Wuxi</td>
<td>Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway</td>
<td>10649600</td>
<td>93014593, 88</td>
<td>125435904</td>
<td>15371</td>
<td>6051,3</td>
</tr>
<tr>
<td>2</td>
<td>Tianhe-2 (MilkyWay-2)</td>
<td>National Super Computer Center in Guangzhou</td>
<td>TH-IIVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P</td>
<td>3120000</td>
<td>33862700</td>
<td>54902400</td>
<td>17808</td>
<td>1901,54</td>
</tr>
<tr>
<td>3</td>
<td>Titan</td>
<td>DOE/SC/Oak Ridge National Laboratory</td>
<td>Cray XK7, Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x</td>
<td>560640</td>
<td>17590000</td>
<td>27112550</td>
<td>8209</td>
<td>2142,77</td>
</tr>
<tr>
<td>4</td>
<td>Sequoia</td>
<td>DOE/NNSA/LLNL</td>
<td>BlueGene/Q, Power BQC 16C 1.60 GHz, Custom</td>
<td>1572864</td>
<td>17173224</td>
<td>20132659,2</td>
<td>7890</td>
<td>2176,58</td>
</tr>
<tr>
<td>5</td>
<td>Cori</td>
<td>DOE/SC/LBNL/NERSC</td>
<td>Cray XC40, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect</td>
<td>622336</td>
<td>14014700</td>
<td>27880653</td>
<td>3939</td>
<td>3557,93</td>
</tr>
<tr>
<td>6</td>
<td>Oakforest-PACS</td>
<td>Joint Center for Advanced High Performance Computing</td>
<td>PRIMERGY CX1640 M1, Intel Xeon Phi 7250 68C 1.4GHz, Intel Omni-Path</td>
<td>556104</td>
<td>13554600</td>
<td>24913459</td>
<td>2718,7</td>
<td>4985,69</td>
</tr>
<tr>
<td>7</td>
<td>RIKEN Advanced Institute for Computational Science (AICS)</td>
<td>K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect</td>
<td></td>
<td>705024</td>
<td>10510000</td>
<td>12659,89</td>
<td>830,18</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Piz Daint</td>
<td>Swiss National Supercomputing Centre (CSCS)</td>
<td>Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100</td>
<td>206720</td>
<td>9779000</td>
<td>15987968</td>
<td>1312</td>
<td>7453,51</td>
</tr>
<tr>
<td>9</td>
<td>Mira</td>
<td>DOE/SC/Argonne National Laboratory</td>
<td>BlueGene/Q, Power BQC 16C 1.60GHz, Custom</td>
<td>786432</td>
<td>8586612</td>
<td>10066330</td>
<td>3945</td>
<td>2176,58</td>
</tr>
<tr>
<td>10</td>
<td>Trinity</td>
<td>DOE/NNSA/LANL/SNL</td>
<td>Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Aries interconnect</td>
<td>301056</td>
<td>8109000</td>
<td>11078861</td>
<td>4232,63</td>
<td>1913,92</td>
</tr>
</tbody>
</table>
Performance Development of HPC over the Last 23 Years from the Top500

- 59.7 GFlop/s in 1994
- 400 MFlop/s
- 1.17 TFlop/s
- 567 PFlop/s
- 286 TFlop/s
- 93 PFlop/s

SUM

N=1

N=500

100 Gflop/s

10 Gflop/s

1 Gflop/s

100 Mflop/s

100 Tflop/s

10 Tflop/s

1 Pflop/s

100 Pflop/s

1 Eflop/s
Supercomputer Performance Road Map

2001: #1 on Top500 is LLNL @ 7 TFlops
AMD Promises 10TFlop Notebooks by 2020
Our origins...Plan Nacional de Investigación

High-performance Computing group @ Computer Architecture Department (UPC)

Relevance

Excellence


CEPBA CIRI BSC
Venimos de muy lejos...
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

BSC-CNS objectives

Supercomputing services to Spanish and EU researchers

R&D in Computer, Life, Earth and Engineering Sciences

PhD programme, technology transfer, public engagement

BSC-CNS is a consortium that includes

Spanish Government 60%

Catalonian Government 30%

Univ. Politècnica de Catalunya (UPC) 10%
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

475 people from 44 countries
*31th of December 2016

Competitive project funding secured (2005 to 2017)

- Europe: 71,9M€
- National: 34 M€
- Companies: 38,9 M€

Total: 144,8 M€

Information compiled 16/01/2017
The MareNostrum 3 Supercomputer

Over $10^{15}$ Floating Point Operations per second

- Nearly 50,000 cores
- 100.8 TB of main memory
- 3 PB of disk storage

70% PRACE, 24% RES, 6% BSC-CNS
The MareNostrum 4 Supercomputer

Total peak performance:

- General Purpose, for current BSC workload:
  - More than 11 Pflops/s
- Emerging Technologies, for evaluation of 2020 Exascale systems:
  - 3 systems, each of more than 0.5 Pflops/s

More than 10 PB of GPFS Elastic Storage System

Network: IB EDR/OPA, Ethernet

Operating System: SuSE

Image of MareNostrum 4 3D sketch with computing nodes, storage, and network connections.
Mission of BSC Scientific Departments

**Computer Sciences**
To influence the way machines are built, programmed and used: computer architecture, programming models, performance tools, Big Data, Artificial Intelligence

**Earth Sciences**
To develop and implement global and regional state-of-the-art models for short-term air quality forecast and long-term climate applications

**Life Sciences**
To understand living organisms by means of theoretical and computational methods (molecular modeling, genomics, proteomics)

**CASE**
To develop scientific and engineering software to efficiently exploit super-computing capabilities (biomedical, geophysics, atmospheric, energy, social and economic simulations)
Design of Superscalar Processors

Decoupled from the software stack

Applications

ISA

Fetch → Decode → Rename → Instruction Window → Wakeup+ select → Register file → Bypass → Data Cache → Register Write → Commit

Programs “decoupled” from hardware

Simple interface Sequential program

ILP
Latency Has Been a Problem from the Beginning... 😞

- Feeding the pipeline with the right instructions:
  - HW/SW trace cache (ICS’99)
  - Prophet/Critic Hybrid Branch Predictor (ISCA’04)

- Locality/reuse
  - Cache Memory with Hybrid Mapping (IASTED87). Victim Cache 😊
  - Dual Data Cache (ICS’95)

- A novel renaming mechanism that boosts software prefetching (ICS’01)
- Virtual-Physical Registers (HPCA’98)
- Kilo Instruction Processors (ISHPC03, HPCA’06, ISCA’08)
... and the Power Wall Appeared Later 😞😞😞

- Better Technologies
- Two-level organization (Locality Exploitation)
  - Register file for Superscalar (ISCA’00)
  - Instruction queues (ICCD’05)
  - Load/Store Queues (ISCA’08)
- Direct Wakeup, Pointer-based Instruction Queue Design (ICCD’04, ICCD’05)
- Content-aware register file (ISCA’09)
- Fuzzy computation (ICS’01, IEEE CAL’02, IEEE-TC’05). Currently known as Approximate Computing 😊
Fuzzy computation

Performance @ Low Power

Fuzzy Computation

Accuracy

Binary systems (bmp)

Compression protocols (jpeg)

This one only used ~85% of the time while consuming ~75% of the power

This image is the original one
Simultaneous Multithreading (SMT)

- Benefits of SMT Processors:
  - Increase core resource utilization
  - Basic pipeline unchanged:
    - Few replicated resources, other shared

Some of our contributions:

- Dynamically Controlled Resource Allocation (MICRO 2004)
- Quality of Service (QoS) in SMTs (IEEE TC 2006)
- Runahead Threads for SMTs (HPCA 2008)
**Time Predictability** (in multicore and SMT processors)

**Definition:**
- Ability to provide a minimum performance to a task
- Requires biasing processor resource allocation

**Where is it required:**
- Increasingly required in handheld/desktop devices
- Also in embedded hard real-time systems (cars, trains, ...)  

**How to achieve it:**
- Controlling how resources are assigned to co-running tasks

**Soft real-time systems**

**Hard real-time systems**
- Deterministic resource ‘securing’ (ISCA 2009)
- Time-Randomised designs (DAC 2014 best paper award)
Vector Architectures... Memory Latency and Power 😊😊😊

- Command Memory Vector (PACT 1998)
  - In-memory computation
- Decoupling Vector Architectures (HPCA 1996)
  - Cray SX1
- Out-of-order Vector Architectures (Micro 1996)
- Multithreaded Vector Architectures (HPCA 1997)
- Vector register-file organization (MICRO 1997)
- Vector Microprocessors (ICS 1999, SPAA 2001)
- Architectures with Short Vectors (PACT 1997, ICS 1998)
  - Tarantula (ISCA 2002), Knights Corner
- High-Speed Buffers Routers (Micro 2003, IEEE TC 2006)
- Vector Architectures for Data-Base (Micro 2012, HPCA2015, ISCA2016)
Statically scheduled VLIW architectures

- Power-efficient FU
  - Clustering
- Widening (MICRO-98)
- \(\mu\text{SIMD}\) and multimedia vector units (ICPP-05)
- Locality-aware
  - Sacks (CONPAR-94)
  - Non-consistent (HPCA95)
  - Two-level hierarchies (MICRO-00)
- Integrated modulo scheduling techniques, register allocation and spilling (MICRO-95, PACT-96, MICRO-96, MICRO-01)
The MultiCore Era

Moore’s Law + Memory Wall + Power Wall

Chip MultiProcessors (CMPs)

POWER4 (2001)

Intel Xeon 7100 (2006)

UltraSPARC T2 (2007)
How Multicores Were Designed at the Beginning?

**IBM Power4 (2001)**
- 2 cores, ST
- 0.7 MB/core L2, 16MB/core L3 (off-chip)
- 115W TDP
- 10GB/s mem BW

**IBM Power7 (2010)**
- 8 cores, SMT4
- 256 KB/core L2
- 16MB/core L3 (on-chip)
- 170W TDP
- 100GB/s mem BW

**IBM Power8 (2014)**
- 12 cores, SMT8
- 512 KB/core L2
- 8MB/core L3 (on-chip)
- 250W TDP
- 410GB/s mem BW
How To Parallelize Future Applications?

- From sequential to parallel codes
- Efficient runs on manycore processors implies handling:
  - Massive amount of cores and available parallelism
  - Heterogeneous systems
    - Same or multiple ISAs
    - Accelerators, specialization
  - Deep and heterogeneous memory hierarchy
    - Non-Uniform Memory Access (NUMA)
    - Multiple address spaces
  - Stringent energy budget
  - Load Balancing

Programmability Wall
Living in the Programming Revolution

Multicores made the interface to leak...

Applications

ISA / API

Parallel application logic + Platform specificites

Parallel hardware with multiple address spaces (hierarchy, transfer), control flows, ...
Vision in the Programming Revolution

Need to decouple again

Applications

PM: High-level, clean, abstract interface

Power to the runtime

ISA / API

Application logic
Arch. independent

General purpose
Single address space

The efforts are focused on **efficiently using** the underlying hardware

Power to the runtime
OmpSs
A forerunner for OpenMP
OmpSs: data-flow execution of sequential programs

```c
void Cholesky( float *A ) {
    int i, j, k;
    for (k=0; k<NT; k++) {
        spotrf (A[k*NT+k]) ;
        for (i=k+1; i<NT; i++)
            strm (A[k*NT+k], A[k*NT+i]);
        // update trailing submatrix
        for (i=k+1; i<NT; i++) {
            for (j=k+1; j<i; j++)
                sgemm( A[k*NT+i], A[k*NT+j], A[j*NT+i]);
            ssyrk (A[k*NT+i], A[i*NT+i]);
        }
    }
}
```

Decouple how we write applications form how they are executed

Write

Clean offloading to hide architectural complexities

Execute

#pragma omp task inout ([TS][TS]A)
void spotrf (float *A);
#pragma omp task input ([TS][TS]A) inout ([TS][TS]C)
void ssyrk (float *A, float *C);
#pragma omp task input ([TS][TS]A,[TS][TS]B) inout ([TS][TS]C)
void sgemm (float *A, float *B, float *C);
#pragma omp task input ([TS][TS]T) inout ([TS][TS]B)
void strm (float *T, float *B);
OmpSs: A Sequential Program ...

```c
void vadd3 (float A[BS], float B[BS],
            float C[BS]);

void scale_add (float sum, float A[BS],
                float B[BS]);

void accum (float A[BS], float *sum);

for (i=0; i<N; i+=BS)  // C=A+B
    vadd3 ( &A[i], &B[i], &C[i]);
...
for (i=0; i<N; i+=BS)  // sum(C[i])
    accum (&C[i], &sum);
...
for (i=0; i<N; i+=BS)  // B=sum*A
    scale_add (sum, &E[i], &B[i]);
...
for (i=0; i<N; i+=BS)  // A=C+D
    vadd3 (&C[i], &D[i], &A[i]);
...
for (i=0; i<N; i+=BS)  // E=G+F
    vadd3 (&G[i], &F[i], &E[i]);
```
#pragma css task input(A, B) output(C)
void vadd3 (float A[BS], float B[BS], float C[BS]);
#pragma css task input(sum, A) inout(B)
void scale_add (float sum, float A[BS], float B[BS]);
#pragma css task input(A) inout(sum)
void accum (float A[BS], float *sum);

for (i=0; i<N; i+=BS) // C=A+B
    vadd3 ( &A[i], &B[i], &C[i]);
...
for (i=0; i<N; i+=BS) // sum(C[i])
    accum (&C[i], &sum);
...
for (i=0; i<N; i+=BS) // B=sum*A
    scale_add (sum, &E[i], &B[i]);
...
for (i=0; i<N; i+=BS) // A=C+D
    vadd3 (&C[i], &D[i], &A[i]);
...
for (i=0; i<N; i+=BS) // E=G+F
    vadd3 (&G[i], &F[i], &E[i]);
Decouple how we write form how it is executed... and Executed in a Data-Flow Model

```c
#pragma css task input(A, B) output(C)
void vadd3 (float A[BS], float B[BS],
            float C[BS]);
#pragma css task input(sum, A) inout(B)
void scale_add (float sum, float A[BS],
               float B[BS]);
#pragma css task input(A) inout(sum)
void accum (float A[BS], float *sum);
```

```c
for (i=0; i<N; i+=BS) // C=A+B
  vadd3 ( &A[i], &B[i], &C[i]);
...
for (i=0; i<N; i+=BS) //sum(C[i])
  accum (&C[i], &sum);
...
for (i=0; i<N; i+=BS) // B=sum*A
  scale_add (sum, &E[i], &B[i]);
...
for (i=0; i<N; i+=BS) // A=C+D
  vadd3 (&C[i], &D[i], &A[i]);
...
for (i=0; i<N; i+=BS) // E=G+F
  vadd3 (&G[i], &F[i], &E[i]);
```

Color/number: a possible order of task execution
OmpSs: Potential of Data Access Info

- Flat global address space seen by programmer
- Flexibility to dynamically traverse dataflow graph “optimizing”
  - Concurrency. Critical path
  - Memory access: data transfers performed by run time
- Opportunities for automatic
  - Prefetch
  - Reuse
  - Eliminate antidependences (rename)
  - Replication management
    - Coherency/consistency handled by the runtime
  - Layout changes
CellSs implementation

Renaming @ Cell

• Experiments on the CellSs (predecessor of OmpSs)
  • Renaming to avoid anti-dependences
    • Eager (similarly done at SS designs)
      • At task instantiation time
    • Lazy (similar to virtual registers)
      • Just before task execution

SMPSs: Stream benchmark reduction in execution time

SMPSs: Jacobi reduction in # remanings

Killed transfers

Main Memory transfers (capacity)

Main memory transfers (cold)

Data Reuse @ Cell

- Experiments on the CellSs
  - Data Reuse
  - Locality arcs in dependence graph

- Good locality but high overhead $\rightarrow$ no time improvement

Reducing Data Movement @ Cell

• Experiments on the CellSs (predecessor of OmpSs)
  • Bypassing / global software cache
  • Distributed implementation
    • @each SPE
    • Using object descriptors managed atomically with specific hardware support (line level LL-SC)

Main memory:
cold

Main memory:
capacity

Global software cache

Local software cache

DMA Reads

P. Belens et al, “Making the Best of Temporal Locality: Just-In-Time Renaming and Lazy Write-Back on the Cell/B.E.” IJHPC 2010
GPUSs implementation

• Architecture implications
  • Large local store O(GB) → large task granularity  ← Good
  • Data transfers: Slow, non overlapped  ← Bad

• Cache management
  • Write-through
  • Write-back

• Run time implementation
  • Powerful main processor and multiple cores
  • Dumb accelerator (not able to perform data transfers, implement software cache,...)
Prefetching @ multiple GPUs

- Improvements in runtime mechanisms (OmpSs + CUDA)
  - Use of multiple streams
  - High asynchrony and overlap (transfers and kernels)
  - Overlap kernels
  - Take overheads out of the critical path

- Improvement in schedulers
  - Late binding of locality aware decisions
  - Propagate priorities

Runtime Aware Architectures

The runtime \textbf{drives} the hardware design

- Applications
- PM: High-level, clean, abstract interface
- Runtime
- ISA / API

- Task based PM annotated by the user
- Data dependencies detected at runtime
- Dynamic scheduling
- “Reuse” architectural ideas under new constraints
## Superscalar vision at Multicore level

### Superscalar World
- Out-of-Order, Kilo-Instruction Processor, Distant Parallelism
- Branch Predictor, Speculation
- Fuzzy Computation
- Dual Data Cache, Sack for VLIW
- Register Renaming, Virtual Regs
- Cache Reuse, Prefetching, Victim C.
- In-memory Computation
- Accelerators, Different ISA’s, SMT
- Critical Path Exploitation
- Resilience

### Multicore World
- Task-based, Data-flow Graph, Dynamic Parallelism
- Tasks Output Prediction, Speculation
- Hybrid Memory Hierarchy, NVM
- Late Task Memory Allocation
- Data Reuse, Prefetching
- In-memory FU’s
- Heterogeneity of Tasks and HW
- Task-criticality
- Resilience

### Load Balancing and Scheduling
- Interconnection Network
- Data Movement
Architecture Proposals in RoMoL

**Runtime Support Unit**
- DVFS
- Light-weight deps tracking
  - Task memoization
  - Reduced data motion

**Cache Hierarchy**
- LM usage
- Coherence
- Eviction policies
- Reductions

**Vectors**
- DB, sorting
- BTrees

**Cluster Interconnect**
- Priority-based arbitration
  - By-pass routing
Runtime Management of Local Memories (LM)

LM Management in OmpSs
- Task inputs and outputs mapped to the LMs
- Runtime manages DMA transfers

8.7% speedup in execution time
14% reduction in power
20% reduction in network-on-chip traffic

Li. Alvarez et al. Transparent Usage of Hybrid on-Chip Memory Hierarchies in Multicores. ISCA 2015.
Li. Alvarez et al Runtime-Guided Management of Scratchpad Memories in Multicore Architectures. PACT 2015
OmpSs in Heterogeneous Systems

Heterogeneous systems
- Big-little processors
- Accelerators
- Hard to program

Task-based programming models can adapt to these scenarios
- Detect tasks in the critical path and run them in fast cores
- Non-critical tasks can run in slower cores
- Assign tasks to the most energy-efficient HW component
- Runtime takes core of balancing the load
- Same performance with less power consumption
Architectural Support for DVFS

Reduce overheads of software solution
- Serialization in DVFS reconfigurations
- User-kernel mode switches

Runtime Support Unit (RSU)
- Power budget
- State of cores
- Criticality of running tasks

Runtime system notifies RSU
- Start task execution
  - Criticality
  - Running core
- End task execution

Same algorithm for DVFS reconfigurations
Architectural Support for DVFS

E. Castillo, CATA: Criticality Aware Task Acceleration for Multicore Processors (IPDPS'16)
TaskSuperscalar (TaskSs) Pipeline

- Hardware design for a distributed task superscalar pipeline frontend (MICRO’10)
  - Can be embedded into any manycore fabric
  - Drive hundreds of threads
  - Work windows of thousands of tasks
  - Fine grain task parallelism

- TaskSs components:
  - Gateway (GW): Allocate resources for task meta-data
  - Object Renaming Table (ORT)
    - Map memory objects to producer tasks
  - Object Versioning Table (OVT)
    - Maintain multiple object versions
  - Task Reservation Stations (TRS)
    - Store and track task in-flight meta-data

- Implementing TaskSs @ Xilinx Zynq

Architectural Support for Task Dependence Management (TDM) with Flexible Software Scheduling

- Task creation is a bottleneck since it involves dependence tracking
- Our hardware proposal (TDM)
  - takes care of dependence tracking
  - exposes scheduling to the SW
- Our results demonstrate that this flexibility allows TDM to beat the state-of-the-art

E. Castillo et al, Architectural Support for Task Dependence Management with Flexible Software Scheduling submitted to MICRO'17
Approximate Task Memoization (ATM)

• Approximate Task Memoization (ATM) aims at eliminating redundant computations.

• ATM leverages runtime system metadata to identify tasks that can be memoized.
  • ATM achieves 1.4x average speedup when only applying memoization techniques (Static ATM).
  • ATM achieves an increased 2.5x average speedup with an average 0.7% accuracy loss with task approximation (Dynamic ATM).

I. Brumar et al, ATM: Approximate Task Memoization in the Runtime System (IPDPS’17)
Exploiting the Task Dependency Graph (TDG) to Reduce Coherence Traffic

- To reduce coherence traffic, the state-of-the-art applies round-robin mechanisms at the runtime level.
- Exploiting the information contained at the TDG level is effective to
  - improve performance
  - dramatically reduce coherence traffic (2.26x reduction with respect to the state-of-the-art).

![State-of-the-art Partition (DEP) Gauss-Seidel TDG]

DEP requires ~200GB of data transfer across a 288 cores system
Exploiting the Task Dependency Graph (TDG) to Reduce Coherence Traffic

• To reduce coherence traffic, the state-of-the-art applies round-robin mechanisms at the runtime level.

• Exploiting the information contained at the TDG level is effective to
  • improve performance
  • dramatically reduce coherence traffic (2.26x reduction with respect to the state-of-the-art).

Graph Algorithms-Driven Partition (RIP-DEP)

RIP-DEP requires ~90GB of data transfer across a 288 cores system

I. Sánchez et al, Reducing Data Movements on Shared Memory Architectures (submitted to SC’17)
Dealing with a New Form Of Heterogeneity

- Manufacturing Variability of CPUs – Different power consumption
- Power variability becomes performance heterogeneity in power constrained environments
- Typical load-balancing may not be sufficient
- Redistributing power and number of active cores among sockets can improve performance
Dynamic Analysis and Exploration

- Statically trying all configurations is not practical
  - Huge overhead (one execution for each configuration)
  - Has to be performed on each node

- Online analysis: Try multiple configurations in a single run.

D. Chasapis et al, Runtime-Guided Mitigation of Manufacturing Variability in Power-Constrained Multi-Socket NUMA Nodes (ICS’16)
Introduction - A New Form Of Heterogeneity

- Platform: 2 x sockets with 12 core Intel Xeon E5-2695v2
- Power variability becomes performance heterogeneity in power constrained environments
Hash Join, Sorting, Aggregation, DBMS

- **Goal:** Vector acceleration of data bases
- **“Real vector” extensions to x86**
  - Pipeline operands to the functional unit (like Cray machines, not like SSE/AVX)
  - Scatter/gather, masking, vector length register
  - Implemented in PTLSim + DRAMSim2
- **Hash join work published in MICRO 2012**
  - 1.94x (large data sets) and 4.56x (cache resident data sets) of speedup for TPC-H
    - Memory bandwidth is the bottleneck
- **Sorting paper published in HPCA 2015**
  - Compare existing vectorized quicksort, bitonic mergesort, radix sort on a consistent platform
- **Propose novel approach (VSR) for vectorizing radix sort with 2 new instructions**
  - Similarity with AVX512-CD instructions (but cannot use Intel’s instructions because the algorithm requires strict ordering)
  - Small CAM
  - 3.4x speedup over next-best vectorised algorithm with the same hardware configuration due to:
    - Transforming strided accesses to unit-stride
    - Eliminating replicated data structures
- **Ongoing work on aggregations**
- **Reduction to a group of values, not a single scalar value ISCA 2016**
  - Building from VSR work
Overlap Communication and Computation

- Hybrid MPI/OmpSs: Linpack example
- Extend asynchronous data-flow execution to outer level
  - Taskify MPI communication primitives
- Automatic lookahead
- Improved performance
- Tolerance to network bandwidth
- Tolerance to OS noise

V. Marjanovic et al, “Overlapping Communication and Computation by using a Hybrid MPI/SMPSs Approach” ICS 2010
Effects on Bandwidth

flattening communication pattern

thus

reducing bandwidth requirements

*simulation on application with ring communication pattern

speedup of 1.6 for the same network bandwidth

for the same execution time

20 times lower needed bandwidth

Related Work

• Rigel Architecture (ISCA 2009)
  • No L1D, non-coherent L2, read-only, private and cluster-shared data
  • Global accesses bypass the L2 and go directly to L3

• SARC Architecture (IEEE MICRO 2010)
  • Throughput-aware architecture
  • TLBs used to access remote LMs and migrate data across LMs

• Runnemede Architecture (HPCA 2013)
  • Coherence islands (SW managed) + Hierarchy of LMs
  • Dataflow execution (codelets)

• Carbon (ISCA 2007)
  • Hardware scheduling for task-based programs

• Holistic run-time parallelism management (ICS 2013)
• Runtime-guided coherence protocols (IPDPS 2014)
RoMoL ... papers

• V. Marjanovic et al., “Effective communication and computation overlap with hybrid MPI/SMPSs.” **PPoPP 2010**
• Y. Etsion et al., “Task Superscalar: An Out-of-Order Task Pipeline.” **MICRO 2010**
• N. Vujic et al., “Automatic Prefetch and Modulo Scheduling Transformations for the Cell BE Architecture.” **IEEE TPDS 2010**
• V. Marjanovic et al., “Overlapping communication and computation by using a hybrid MPI/SMPSs approach.” **ICS 2010**
• T. Hayes et al., “Vector Extensions for Decision Support DBMS Acceleration”. **MICRO 2012**
• L. Alvarez, et al., “Hardware-software coherence protocol for the coexistence of caches and local memories.” **SC 2012**
• M. Valero et al., “Runtime-Aware Architectures: A First Approach”. **SuperFRI 2014**
RoMoL ... papers

- M. Casas et al., “Runtime-Aware Architectures”. **Euro-Par 2015.**
- L. Alvarez et al., “Run-Time Guided Management of Scratchpad Memories in Multicore Architectures”. **PACT 2015**
- L. Jaulmes et al., “Exploiting Asynchrony from Exact Forward Recoveries for DUE in Iterative Solvers”. **SC 2015**
- D. Chasapis et al., “PARSECSs: Evaluating the Impact of Task Parallelism in the PARSEC Benchmark Suite.” **ACM TACO 2016.**
RoMoL ... papers


• D. Chasapis et al., “Runtime-Guided Mitigation of Manufacturing Variability in Power-Constrained Multi-Socket NUMA Nodes.” ICS 2016

• P. Caheny et al., “Reducing cache coherence traffic with hierarchical directory cache and NUMA-aware runtime scheduling.” PACT 2016


• I. Brumar et al., “ATM: Approximate Task Memoization in the Runtime System.” IPDPS 2017

• K. Chronaki et al., “Task Scheduling Techniques for Asymmetric Multi-Core Systems.” IEEE TPDS 2017

• C. Ortega et al., “libPRISM: An Intelligent Adaptation of Prefetch and SMT Levels.” ICS 2017
Roadmaps to Exaflop

From K computer... to Post K with domestic technology.

From Tianhe-2... to Tianhe-2A with domestic technology.

IPCEI on HPC

From the PPP for HPC... to future PRACE systems... with domestic technology
HPC is a global competition

“The country with the strongest computing capability will host the world’s next scientific breakthroughs”.

US House Science, Space and Technology Committee Chairman Lamar Smith (R-TX)

“Our goal is for Europe to become one of the top 3 world leaders in high-performance computing by 2020”.

European Commission President Jean-Claude Juncker (27 October 2015)

“Europe can develop an exascale machine with ARM technology. Maybe we need an consortium for HPC and Big Data”.

Seymour Cray Award Ceremony Nov. 2015 Mateo Valero
HPC: a disruptive technology for Industry

“The transformational impact of excellent science in research and innovation”

Final plenary panel at ICT - Innovate, Connect, Transform conference, 22 Oct 2015, Lisbon.

“...Europe has a unique opportunity to act and invest in the development and deployment of High Performance Computing (HPC) technology, Big Data and applications to ensure the competitiveness of its research and its industries.”

Günther Oettinger, Digital Economy & Society Commissioner
BSC and the EC

Final plenary panel at ICT - Innovate, Connect, Transform”m conference, 22 October 2015 Lisbon, Portugal.

"Europe needs to develop an entire domestic exascale stack from the processor all the way to the system and application software"

Mateo Valero, Director of Barcelona Supercomputing Center

the transformational impact of excellent science in research and innovation
Mont-Blanc HPC Stack for ARM

Industrial applications

Applications

System software

Hardware
BSC Accelerator

512 RiscV cores in 64 clusters, 16GF/core:
- 8TF
- 64GB @ 4TB/s

4 HBM stacks (16GB, 1TB/s each):
- 64GB @ 4TB/s

16 custom SCM/Flash channels (1TB, 25GB/s each):
- 16TB @ 0.4TB/s

RISC-V ISA
- Vector Unit
  - 2048b vector
  - 512b alu (4clk/op)

1 GHz @ Vmin

OOO

4w Fetch
- 64KB I$
- Decoupled I$/BP
- 2 level BP
- Loop Stream Detector

4w Rename/Retire
D$
- 64KB
- 64B/line
- 128 in-flight misses
- Hardware prefetch

1MB L2 per core

D$ to L2
- 1x512b read
- 1x512b write

L2 to mesh
- 1x512b read
- 1x512b write

Cluster holds snoop filter
HPC European strategy & Innovation

A window of opportunity is open:
• Basic industrial and scientific know-how is available
• Excellent funding opportunities exist in H2020 at European level and in the member state structural funds

It’s time to invest in large Flagship projects for HPC to gain critical mass

Do we need an type consortium for HPC and Big Data?

HPC European strategy & Innovation

Current infrastructure sagging under its own weight

2013
- 98,000 tweets
- 23,148 apps downloaded
- 400,710 ad requests
- 2000 lyrics played on Tunewiki
- 1,500 pings sent on PingMe
- 208,333 minutes Angry Birds played

By 2020
- 30 Billion [1]
- 40 Trillion GB [2]
- 10 Million [3]
- ... for 8 Billion [4]

Internet of Things

Pervasive Connectivity | Smart Device Expansion | Explosion of Information

(1) IDC Directions 2013: Why the Datacenter of the Future Will Leverage a Converged Infrastructure, March 2013, Matt Eastwood; (2) & (3) IDC Predictions 2012: Competing for 2020, Document 231720, December 2011, Frank Gens; (4) http://en.wikipedia.org
The Data Deluge

2005
0.1ZB

2010
1.2ZB

2012
2.8ZB

2015
8.5ZB

2020
40ZB*

(figure exceeds prior forecasts by 5 ZBs)

* Source: IDC

Prof. Mateo Valero – Big Data
This will take us beyond our decimal system. Geopbyte.

This will be our digital universe tomorrow.

Yottabyte

This is our digital universe today = 250 trillion of DVDs.

Brontobyte

1 EB of data is created on the internet each day = 250 million DVDs worth of information. The proposed Square Kilometer Array telescope will generate an EB of data per day.

Petabyte

The CERN Large Hadron Collider generates 1PB per second.

Zettabyte

1.3 ZB of network traffic by 2016.

Exabyte

10^18

Terabyte

10^12

Gigabyte

10^9

Megabyte

10^6

How big is big?

Saganbyte, Jotabyte,…

Prof. Mateo Valero – Big Data
Last year one of the most computer-intensive scientific experiments ever undertaken confirmed Peter Higgs and François Englert’s theory by making the Higgs boson – the so-called “God particle” – in an $8bn atom smasher, the Large Hadron Collider at Cern outside Geneva.

“the LHC produces 600 TB/sec... and after filtering needs to store 25 PB/year”... 15 million sensors....
Big Data in Biology

As they grapple with increasingly large data sets, biologists and computer scientists uncover new horizons.

**By Vivien Riber**

Biologists are creating the big data club. With the advent of high-throughput protocols, life sciences are starting to grapple with massive data sets, uncovering challenges in the handling, processing, and accessing information that were once the domains of astronomers and high-energy physicists.

With every passing year, they turn more often to big data to probe everything from the regulation of genes and the evolution of genomes to why certain signs ¬exist, what mechanisms drive human body activities, and how the genetic make-up of different cancer subtypes influences the cancer patient's fate. The European Molecular Biology Laboratory in Heidelberg, Germany, part of the European Molecular Biology Laboratory (EMBL) and one of the world's largest biomedical research institutions, generates more than 20 petabytes of data per year - a data set the size of the entire Encyclopedia Britannica in text. How can big data be harnessed to answer biological questions? What do we need to do to maximize data to yield research answers? As prices drop for high-throughput technologies, how are we capitalizing on these advances?
Sequencing Costs

Source: National Human Genome Research Institute (NHGRI)
http://www.genome.gov/sequencingcosts/

(1) "Cost per Megabase of DNA Sequence" — the cost of determining one megabase (Mb; a million bases) of DNA sequence of a specified quality
(2) "Cost per Genome" - the cost of sequencing a human-sized genome. For each, a graph is provided showing the data since 2001

In both graphs, the data from 2001 through October 2007 represent the costs of generating DNA sequence using Sanger-based chemistries and capillary-based instruments (‘first generation’ sequencing platforms). Beginning in January 2008, the data represent the costs of generating DNA sequence using ‘second-generation’ (or ‘next-generation’) sequencing platforms. The change in instruments represents the rapid evolution of DNA sequencing technologies that has occurred in recent years.
Cognitive Computing

Acuerdo de colaboración para promover conjuntamente el desarrollo de sistemas avanzados de “deep learning” con aplicaciones a los servicios bancarios
Example: Cognitive Computing is already in business

• In 2011 IBM Watson computer defeated two of Jeopardy’s greatest champions

Since then, Watson supercomputer has become 24 times faster and smarter, 90% smaller, with a 2,400% improvement in performance

Watson Group has collaborated with partners to build 6,000 apps
Neural Networks

- Computational model in computer science based on a collection of simple neural units.
- Each neural unit is connected to many others
  - The strengths of these connections is expressed in terms of weights.
  - Neural units compute summation functions
- NN are self-learning and can be trained
- NN are particularly good in feature detection.
- In practice, NN can be expressed in terms of matrix-matrix multiplications.
RECONOCIMIENTO DE IMÁGENES

Ratio de Error [%]

2010 2011 2012 2013 2014

27 26 12 11 6

Introducción de las redes neuronales a gran escala
650.000 neuronas

5.000.000 neuronas

ERROR

HUMANO

SC-2017-SLC
Dios los cría y la IA+HPC los junta....
Introducción de las redes neuronales a gran escala
# BSC strategy for Artificial Intelligence

## Projects with public/private institutions and companies

<table>
<thead>
<tr>
<th>Precision medicine</th>
<th>Other domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genomic Analytics</td>
<td>Social &amp; Personal Data</td>
</tr>
<tr>
<td>Text Analytics</td>
<td>Industrial CASE apps</td>
</tr>
<tr>
<td>Medical Imaging</td>
<td>Earth Sciences</td>
</tr>
<tr>
<td>Organ simulation</td>
<td></td>
</tr>
</tbody>
</table>

### Data models and algorithms
- (approximate computing -- reduced precision, adaptive layers, DL/Graph Analytics, ...)

### Programming models and runtimes
- (PyCOMPSs, TIRAMISU, interoperability current approaches)

### Hw acceleration of DL workloads
- (novel architectures for NN, FPGA acceleration)

### Data platforms + standards
NVIDIA Tesla P4 and P40 GPU’s (2016)

• Tesla P4
  • # CUDA cores: 2560 @ 1063MHz
  • Peak single precision: 5.5TFLOPS
  • Peak INT8: 22 TOPS
  • Low precision: 8-bit dot-product with 32-bit accumulate
  • VRAM: 8 GB GDDR5 @ 192 GB/s
  • TDP: ~75W

• Tesla P40
  • # CUDA cores: 2560 @ 1531MHz
  • Peak single precision: 12.0TFLOPS
  • Peak INT8: 47 TOPS
  • Low precision: 8-bit dot-product with 32-bit accumulate
  • VRAM: 24 GB GDDR5 @ 346GB/s
  • TDP: ~250W

Source: NVIDIA

- 34 GB/s off-chip memory
- 28MB on-chip memory
- Frequency 700MHz
- TDP 75W
- Matrix Multiply Unit
  - 256x256 MAC Units
  - 8-bit multiply and adds
  - 32-bit accumulators
- Peak Throughput: 92 TOPS/s
- Power Efficiency: 132 GOPS/W
- GPUs for training, TPUs for inference
- Gameplay to beat the World Go champion
- Internally used at google for Streetview and
- Rankbrain search optimizer

Source: google
Nervana’s Lake Crest Deep Learning Architecture (2017)

- The Lake Crest chip will operate as a Xeon Co-processor.
- Tensor-based (i.e. dense linear algebra computations)
- 4 8GB HBM2 at the same chip interposer@1TB/s
  - Each HBM has its own memory controller
- 12 Inter-Chip Links (ICL) 20x faster than PCI
- 12 Computing Nodes featuring several cores
- Intel's new "Flexpoint" architecture within the Nodes
  - Flexpoint enables 10x ILP increase and low power consumption
Quantitative Analysis of Deep Learning Architectures

- Each N-neuron Hidden Layer (HL) requires a NxN GEMM
- 2D NxN Systolic Array carries out NxN GEMM in 2N+1 cycles.

![Graphs showing performance metrics for different matrix sizes and frequencies.](image)
Quantitative Analysis of Deep Learning Architectures

- Each N-neuron Hidden Layer (HL) requires a NxN GEMM
- 2D NxN Systolic Array carries out NxN GFMM in 2N+1 cycles.

![Graph showing GB/s vs Matrix Size and OP/s vs Matrix Size](image)

- HL of ~1024 neurons can identify simple images
  - 28x28 pixel images
  - Each image contains a digit 0-9
Quantitative Analysis of Deep Learning Architectures

- Each N-neuron Hidden Layer (HL) requires a NxN GEMM

- 2D NxN Systolic Array carries out NxN GEMM in 2N+1 cycles.

- HL of ~4096 neurons can identify images containing a single concept
  - 32x32 pixel images
  - Each image is classified by categories like “ship”, “cat” or “deer”.
Quantitative Analysis of Deep Learning Architectures

• Each N-neuron Hidden Layer (HL) requires a NxN GEMM

• 2D NxN Systolic Array carries out NxN GEMM in 2N+1 cycles.

Lake Crest’s mem BW (~TB/s) targets very large HL with O(10,000-100,000) neurons

These NN are used for complex image analysis
BSC Proposal for Deep Learning

16 2D-systolic arrays 4096x4096@1GHz: 134TOP/s
4 HBM stacks (16GB@1TB/s each): 64 GB @ 4TB/s
DDR5 SDRAM (384GB@180GB/s): 384GB @ 0.18TB/s
Human Brain Project

- 10-year, 1000M€ FET flagship project
- Goal: to pull together all existing knowledge about the human brain and to reconstruct the brain in supercomputer based models and simulations.

Expected outcomes: new treatments for brain disease and new brain-like computing technologies

BSC role: Provision and optimisation of programming models to allow simulations to be developed efficiently

MareNostrum part of the HPC platform for simulations
View from Europe: SpiNNaker machine

- HBP platform
  - 500,000 cores
  - 6 cabinets (including server)
- Launch
  - 30 March 2016
IBM TrueNorth Processor

- 64*64=4096 cores
- 256 neurons/core, 64K synapses/core
- 104Kb/core memory
  - 65Kb for synapse states
  - 32Kb for neuron states/parameters
  - 6Kb for router destination addresses
  - 1Kb for axonal delays

- 20mW/cm² power density
- 72mW at 0.75V

- 46 Billion SOPS/Watt (Synoptic Operations Per Second) typ.
- 400 Billion SOPS/Watt max.
- Compared to SoA supercomputer at 4.5 Billion FLOPS/Watt

Source: Science magazine
View from Europe: Heidelberg HICANN

- Wafer-scale analogue neuromorphic system

- 8” 180nm wafer:
  - 200,000 neurons
  - 50M synapses
  - $10^4$x faster than biology
Quantum Computing: Brave New World of post-Moore architecture

Quantum Processors

- Dwave
- IBM
- Microsoft
- Google
- View from Europe: Delft University Prototypes
D-Wave Quantum Processor

- Environment colder than space
- Leverages superconducting quantum effect
- 1000 qbits, 128K josephson junctions
- Installed at NSA, Google, UCSB
- $10^8$X faster than Quantum Monte Carlo Algorithm on a single core*

IBM

Building “Universal Quantum Computer”

Developed a Quantum Computing API to make developing quantum applications easier

Promotes experimentation on publicly available 5-qbit quantum processor
Microsoft and Google

Microsoft is looking into topological quantum computing in their global “Station Q” research consortium.

Microsoft has “Quarc” lab working actively in quantum computer architecture in Redmond.

Google manufactured a 9-bit Quantum Computer in their Quantum AI Lab.

Google ambition is to produce a viable quantum computer in the next five years*.

View from Europe: Delft Quantum Prototypes

- 50M Euro grant from Intel
- Building hybrid CMOS/Quantum processor
- Doing algorithms, compilers, architecture

THANK YOU!

www.bsc.es