The DEEP Project

Pursuing cluster-computing in the many-core era

N. Eicker, Th. Lipper, E. Suarez · Jülich Supercomputing Centre
Th. Moschny · ParTec Cluster Competence Center

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under Grant Agreement n° 287530

HUCAA’13
2nd International Workshop on Heterogeneous and Unconventional Cluster Architectures and Applications
October 1st 2013 - Lyon, France
In conjunction with the 42nd International Conference on Parallel Processing (ICPP 2013)
Evolution

Multi-core ceases scaling
Many-core is more promising
ExaScale Challenges

- Multi-PetaFlop ($10^{15}$) systems are up an running
  - Tianhe-2, Titan, Sequoia, K, JuQUEEN
  - Sustained PetaFlop for broader range of applications to come
- History shows: each scale (factor 1000) takes ~10 years
- Look at problems to expect for next step: ExaFlop ($10^{18}$)
  - Power consumption (are ~100 MW acceptable?)
  - Resiliency
  - How to program such beast
    - Programming models
    - Do current algorithms still work out?
  - Applications
Technology Scaling

Moore’s Law

# of transistors / area doubles in 1.5 years

→ in 10 years: \(2^{6.6} = 100\)

Holds since 40 years based on silicon technology

Meuer’s Law

Supercomputer Performance increases by factor 1000 in 10 years (so far)
**Rationale**

- Can the next generation cluster computers compete with proprietary solutions like BlueGene or Cray?
  - BlueGene /P → /Q gives factor 20 in compute speed at the same energy envelope and costs in 4 years
  - Cray is more dependent on processor development

- Standard processor speed will increase by about a factor of 4 to at most 8 in 4 years…
  - Clusters need to utilize accelerators
    - Current accelerators are not parallelized on the node-level

- Integrated processors (for HPC) expected in 2015…
- Heterogeneous Architectures will be there in the future
Heterogeneous Clusters

- Flat topology
- Simple management of resources
- Static assignment of accelerators to CPUs
- Accelerators cannot act autonomously
Accelerated Cluster vs. Cluster of Accelerators

**Cluster with Accelerators**
- Accelerator needs a host CPU
- Static assignment (host CPU with 1 or more acc.)
- Communication so far via main memory
- PCIe bus turns out to be a bottleneck
- Requires explicit GPU programming (Cuda, OpenCL, etc.)

**Cluster of Accelerators only**
- Node consists of Accelerator directly connected to network
- (Only possible with MIC today)
- Static and dynamical assignment possible
- Concept can be adapted to concurrency levels
Alternative Integration

- Go for more capable accelerators (e.g. MIC)
- Attach all nodes to a low-latency fabric
- All nodes might act autonomously
- Dynamical assignment of cluster-nodes and accelerators
  - IB can be assumed as fast as PCIe besides latency
- Ability to off-load more complex (including parallel) kernels
  - communication between CPU and Accelerator less frequently
  - larger messages i.e. less sensitive to latency
Application's Scalability

- Only few applications capable to scale to $O(300k)$ cores
  - Sparse matrix-vector codes
  - Highly regular communication patterns
  - Well suited for BG/P

- Most applications are more complex
  - Complicated communication patterns
  - Less capable to exploit accelerators

- How to map different requirements to most suited hardware
  - Heterogeneity might be a benefit
  - We need better programming models
Cluster-Booster Architecture

Keep flexibility due to IB between cluster-nodes and booster-nodes
Complex kernels to be offloaded expected to have regular communication patterns
Kernels relieve pressure on CPU to Acc. communication
DEEP project

16 partners from 8 countries:
3 PRACE Hosting Members
5 industry partners

Start: 1\textsuperscript{st} Dec 2011
Duration: 3 years
Budget: 18.5 M€ (8.03 M€ funded by EU)
Dynamical Exascale Entry Platform (DEEP)

- EU ICT-2010.9.13 call
- Proposes to develop a novel, Exascale-enabling supercomputing platform
- Along with the optimization of a set of representative grand-challenge codes simulating applications
- Concept based on an advanced multi-core cluster system with InfiniBand complemented by a booster of Intel many-core MIC processors connected through a Terabit EXTOLL network
- 16 partners from 8 countries
- 3 e-Infrastructure providers
  - BSC
  - LRZ
  - FZJ (Coordinator)
- 5 industry partners
  - EuroTech
  - University of Heidelberg / Extoll
  - Intel GmbH
  - Mellanox Technologies
  - ParTec Cluster Competence Center
- 8 application partners
  - German Research School
  - École polytechnique fédérale de Lausanne
  - Katholieke Universiteit Leuven
  - CERFACS
  - Cyprus Institute
  - University of Regensburg
  - CINECA
  - CGGVerita
Partners
DEEP Architecture

[Diagram showing the DEEP architecture with Cluster Nodes, Booster Interface Nodes, and Booster Nodes connected by InfiniBand.]

EXTOLL 3D Torus Network
• Important features for DEEP:
  – High performance
  – Sufficient memory bandwidth
  – Possibility to directly attach a network
  – Ability to run general purpose codes (MPI-library)
  – Autonomous operation (with EXTOLL)
  – Energy efficient: 5 GFlop/W
  – Direct water cooling possible
  – Extensible software stack: network drivers
Relevant features for DEEP

- Low latency, high bandwidth
- RMA engine for remote memory access, bulk data transfer
- VELO communication engine (zero-copy MPI)
- SMFU engine for bridging to InfiniBand
- 6 links for 3D torus topology
- 7th link for general devices
- Built-in PCIe root-port
- RAS features: CRC/ECC protection, link level retransmission
- Many status & control registers
- Access from host, via I2C bus or over EXTOLL
Hardware Integration
Positioning DEEP

- IBM Power 4
  9 TFlop/s

- IBM Power 6
  9 TFlop/s

- INTEL Nehalem cluster
  300 Tflop/s

- IBM Blue Gene/P
  223 TFlop/s

- IBM Blue Gene/P
  1,0 PFlop/s

- INTEL Nehalem cluster
  300 Tflop/s

- IBM Blue Gene/P
  5.9 PFlop/s

DEEP System

Cluster

InfiniBand

BI

BN

BN

BN

EXTOLL

BI

BN

BN

BN

BN

Cluster

Booster

Low - medium scalable architecture

Highly scalable architecture
Programming Model
Software Architecture

Application

Less scalable code parts

Highly scalable code parts

OmpSs Offload Abstraction

ParaStation Global MPI

ParaStation Cluster MPI

Resource Management

Cluster-Booster communication

Low-Level InfiniBand® Communication

Low-Level EXTOLL Communication

OmpSs Compiler

Intel® Compiler for Xeon®

DEEP Cluster

OmpSs Compiler

Intel® Compiler for MIC

DEEP Booster
Application Startup in Detail

- Application's main()-part runs on Cluster-nodes (CN) only
- Resources managed statically or dynamically
- OmpSs acts as an abstraction layer
- Actual spawn done via global MPI

- Spawn is a collective operation of Cluster-processes
- Highly scalable code-parts (HSCP) utilize multiple Booster-nodes (BN)
OmpSs: tasks, dependencies, heterogeneity

Decouple how we write (think sequential) from how it is executed

void Cholesky( float *A[NT] ) {
    int i, j, k;
    for (k=0; k<NT; k++) {
        spotrf (A[k][k]) ;
        for (i=k+1; i<NT; i++)
            strsm (A[k][k], A[k][i]);
        for (i=k+1; i<NT; i++) {
            for (j=k+1; j<i; j++)
                sgemm( A[k][i], A[k][j], A[j][i]);
            ssyrk (A[k][i], A[i][i]);
        }
    }
}

#pragma omp task inout ([TS][TS]A)
void spotrf (float *A);
#pragma omp task input ([TS][TS]T) inout ([TS][TS]B)
void strsm (float *T, float *B);
#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]A) inout ([TS][TS]C)
void ssyrk (float *A, float *C);

Decouple how we write (think sequential) from how it is executed
Heterogeneous global MPI

Application

Less scalable code parts

Highly scalable code parts

OmpSs Offload Abstraction

ParaStation Global MPI

ParaStation Cluster MPI

Resource Management

Cluster-Booster communication

ParaStation Booster MPI

Low-Level InfiniBand® Communication

Low-Level EXTOLL Communication

OmpSs Compiler

Intel® Compiler for Xeon®

CN CN CN CN CN

DEEP Cluster

OmpSs Compiler

Intel® Compiler for MIC

BI BI BN BN BN BN

DEEP Booster
Offloading Invocation

- There is an existing MPI program
- Highly scalable kernels are identified
- Adapt to the Cluster-Booster Architecture
- How to specify …
  - which code is to run on the Booster nodes
  - where on the Booster it should run
  - which data is to be copied between Cluster and Booster before/after a Booster code part is executed
  - how the data layout has to be transformed
Low-Level Offloading Semantics

- **Basic idea: Provide a “Global MPI”**
  - Connect Cluster MPI and Booster MPI via MPI_Comm_spawn()
  - Startup mechanism for Booster code parts
  - The children have their own MPI_COMM_WORLD (different from the parents' one)
MPI Process Creation

MPI_Comm_spawn(
  → command,       // command (string)
  → argv,          // arguments (string[])
  → maxprocs,      // # of processes to start (int)
  → info,          // key-value pairs (handle)
  → root,          // rank of root process
                    //   (int, for prev. args)
  → comm,          // parents' communicator
                    //   (handle)
  ← intercomm,     // intercommunicator (handle)
  ← errorcodes     // one code per process (int[])
)
Programming Model

• Based on MPI
  – Parallel programming model well-known to application developers
  – High-performance implementations available
  – Used by Cluster code as well as by the Booster code parts

• ParaStation MPI
  – Works “out of the box” on the Cluster part
  – Currently ported to the Booster part
  – Integrates well with the ParaStation Cluster Management Software
Global MPI

Cluster

Booster Interface

Booster

ParaStation MPI

Infiniband

MPI_Comm_spawn

Cluster Booster Protocol

Extoll
OmpSs Offload Abstraction
OmpSs Offload Abstraction

```c
int main(int argc, char *argv[]){
    /*...*/
    for(int i=0; i<3; i++){
        #pragma target device (comm:size*rank+i) copy_deps
        #pragma omp task input(...) output(...)
        foo_mpi(i, ...);}
}
```
Take aways

- Exascale poses severe challenges
  - Energy, Resiliency, Scalability, Programmability
  - Have to face more and huger levels of parallelism
  - Computing will become (even more) heterogeneous

- Some new ideas are around → DEEP
  - tries to handle heterogeneity in an innovate way
  - allows to map application's levels of scalability onto hardware
  - follows new approaches for the programming paradigm

- More info:  [http://www.deep-project.eu](http://www.deep-project.eu)