Programming model and application porting to the Dynamical Exascale Entry Platform (DEEP)

EASC 2013
April 10th, Edinburgh
Damián A. Mallón

The research leading to these results has received funding from the European Community’s Seventh Framework Programme (FP7/2007-2013) under Grant Agreement nº 287530
DEEP Architecture
Programming Model
Applications
  - Climate Simulation (CYI)
  - CFD (CERFACS)
  - Superconductivity (CINECA)
Conclusions
DEEP is a collaborative effort with contribution from many partners

Contribution to the work here presented

– Jülich Supercomputing Centre
– University of Heidelberg/EXTOLL
– Par-Tec
– Barcelona Supercomputing Centre
– Cyprus Institute
– CERFACS
– CINECA
DEEP Architecture

Constellation Systems

Intel Cluster Systems (Juropa)

Graphic-card Accelerated Clusters

HPC Cluster Architectures

IBMP Blue Gene/L

IBMP Blue Gene/P

IBMP Blue Gene/Q

Extremely Scalable Architectures
DEEP Architecture

- **Cluster Nodes**
  - 128 Nodes
  - 2x Sandy Bridge processors
  - InfiniBand network
    - Fat tree
  - Connected to I/O

- **Booster Nodes**
  - 512 Nodes
  - 1x Xeon Phi
  - EXTOLL network
    - 3D torus

- **Cluster-Booster Interface Nodes**
  - 32 Nodes
  - InfiniBand and EXTOLL
# DEEP Architecture

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>EXTOLL VENTOUX</td>
<td>Xilinx Virtex6 FPGA</td>
<td>200 MHz</td>
<td>16 Gb/s</td>
<td>1.2 µs</td>
<td>Up to 1.4 GB/s</td>
<td>~ 25 millions</td>
</tr>
<tr>
<td>EXTOLL TOURMALET</td>
<td>65 nm ASIC</td>
<td>750 MHz</td>
<td>120 Gb/s</td>
<td>0.6 µs</td>
<td>Up to 10 GB/s</td>
<td>~ 100 millions</td>
</tr>
<tr>
<td>IB FDR*</td>
<td>45 nm ASIC</td>
<td>Unknown</td>
<td>56 Gb/s</td>
<td>0.8 µs</td>
<td>Not measured</td>
<td>Not measured</td>
</tr>
<tr>
<td>Typical 1GE</td>
<td>ASIC based</td>
<td>e.g. 125 MHz</td>
<td>1 Gb/s</td>
<td>e.g. 40 µs</td>
<td>0.11 GB/s</td>
<td>~0.5 millions</td>
</tr>
<tr>
<td>10GE</td>
<td>ASIC based</td>
<td>~125 to 312 MHz</td>
<td>12.5 Gb/s</td>
<td>12.5 µs</td>
<td>1.2 GB/s</td>
<td>&lt; 2.5 millions</td>
</tr>
<tr>
<td>Cray Gemini</td>
<td>90 nm ASIC</td>
<td>650/800 MHz</td>
<td>75 Gb/s</td>
<td>1.5 µs</td>
<td>Up to 5.9 GB/s</td>
<td>~ 2 millions</td>
</tr>
<tr>
<td>Tianhe-1a</td>
<td>ASIC based</td>
<td>Unknown</td>
<td>80 Gb/s</td>
<td>2.5 µs</td>
<td>Up to 6.34 GB/s</td>
<td>~ 1-3 millions</td>
</tr>
<tr>
<td>TOFU (K-Computer)</td>
<td>65 nm ASIC</td>
<td>312.5 MHz</td>
<td>50 Gb/s</td>
<td>1.5 µs</td>
<td>Up to 4.76 GB/s</td>
<td>&gt; 8 millions</td>
</tr>
</tbody>
</table>

* *Mellanox literature data*
Programming Model

- Less scalable code parts
- Highly scalable code parts

OmpSs Offload Abstraction

ParaStation Global MPI
- Resource Management
- Cluster-Booster communication

ParaStation Cluster MPI
- ParaStation Booster MPI

Low-Level InfiniBand® Communication

Low-Level EXTOLL Communication

OmpSs Compiler
- Intel® Compiler for Xeon®
- DEEP Cluster

OmpSs Compiler
- Intel® Compiler for MIC
- DEEP Booster
Programming Model

Cluster

Booster

Data set

Local data set for each node or MPI rank
“Global MPI”
- Connect Cluster MPI and Booster MPI via MPI_Comm_spawn()
- Startup mechanism for Booster code parts
- P2P communication mechanism via intercommunicator
Decouple how we write (think sequential) from how it is executed

```c
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++)
            strm (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]);
            ssysr (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 strm (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 ssysr (float *A, float *C);
```
Approach

- MPI+OpenMP/OmpSs @ Cluster
- MPI+OpenMP/OmpSs @ Booster
- MPI+OpenMP/OmpSs @ whole System (Global MPI)
“Collective” offload approach

Collective allocation of Booster Nodes

- Define resources and communication name space

User controlled placement of tasks

- Tasks can use MPI between them within allocated communicator

Standard MPI and OmpSs elsewhere

- i.e.: CN ↔ BN transfers are asynchrony handled by standard OmpSs

```c
int DEEP_Booster_alloc( MPI_Comm cluster_comm,
                          int booster_nodes, ...,
                          MPI_Comm *intercomm);
```

```c
#define pragma omp target device (intercomm:rank)
```
int main(int argc, char *argv[]){
    MPI_Init(...);MPI_Comm_rank(...);MPI_Comm_size(...);
    MPI_Comm comm;
    DEEP_Booster_alloc(MPI_COMM_WORLD, size*3, size*3, &comm);
    for(int i=0; i<3; i++){
        #pragma target device (comm:size*rank+i) copy_deps
        #pragma omp task input(…) output(…)
        foo_mpi(i,…);
    }
}
Top-Level abstraction: OmpSs
- **Hide the implementation details** from the application developer
- Developer **annotates** the code to specify offloading semantics

OmpSs allows to specify …
- Booster topology (torus or grid, size)
- Data distribution

OmpSs compiler
- Reads annotations, generates code calling the runtime
- Generates (separate) executables for Cluster and Booster

OmpSs runtime
- Initiates kernels in the Booster
- Handles data transfer between Cluster and Booster
Applications

- Climate Simulation (CYI)
- Computational Fluid Dynamics (CERFACS)
- High Temperature Superconductivity (CINECA)
- Seismic Imaging (CGGVeritas)
- Space Weather (KULeuven)
- Brain Simulation (EPFL)
Climate Simulation (CYI)

- Partner: CYI
- EMAC (ECHAM/MESSy Chemistry)
  - ECHAM provides a comprehensive general circulation model (GCM) of the atmosphere
  - MESSy (Modular Earth Submodel System) couples several physicochemical processes to it.
- Fortran 95
Climate Simulation (CYI)

- Two coupled models:
  - ECHAM (Atmospheric)
  - MESSy (Physicochemical interactions)
  - EMAC = ECHAM/MESSy Atmospheric Chemistry
Climate Simulation (CYI)

- Total Execution Time:
  - Computation
  - Communication

Phases of the Application:
- Total
- MECCA
- MESSy global end
- Others

Percentage of the total execution time
Climate Simulation (CYI)

Cluster:
- 3D transpositions
- Inverse Legendre transforms
- 3D transpositions
- Inverse Fourier transforms
- 3D transpositions
- Transport core calculations, 4D transpositions

Grid point calculations, local, physics, chemistry:
- Direct Fourier transforms
- 3D transpositions
- Direct Legendre transforms
- 3D transpositions
- Spectral integration, non-local, meteorology

Plugin:
- MESSy
- MECCA

Booster:
CFD (CERFACS)

- Partner: CERFACS
- AVBP
- 3D Compressible Navier-Stokes solver
- Fortran + C
• Application structure
  – Phases
    • Read data
    • Decompose
    • Compute loop
      – Write snapshot every X cycles
  – The initial and I/O phases used to follow a master-slave approach
    • They have been optimized for scalability recently
  – Domain decomposition is done in parallel
  – I/O as well
CFD (CERFACS)

Iteration loop

- Compute
  - Thermodynamics
  - Gradients
  - Gas turbulent model
  - Liquid turbulent model
  - Time step
  - Fluxes
  - Residuals
  - Chemical source terms

- Averaging

- Store w fields

- Computes and outputs boundary monitoring data

- Extract and output temporal monitoring data

- Impose boundary targets

- Update w fields
CFD (CERFACS)

- Trace of a 16 core simulation

- Preprocessing
  - collective load data
  - element graph & parmetis

- Temporal loop
  - read & metric
  - iteration
  - synchronization of partitions
• Trace analysis (old version)
  – Computation takes ~50% of the time
  – Synchronization and boundary exchange takes ~20% of the time
CFD (CERFACS)

**Cluster**
- Iteration loop
- Compute
- Averaging
- Store w fields
- Computes and outputs boundary monitoring data
- Extract and output temporal monitoring data

**Booster**
- Compute:
  - Thermodynamics
  - Gradients
  - Gas turbulent model
  - Liquid turbulent model
  - Time step
  - Fluxes
  - Residuals
  - Chemical source terms
- Impose boundary targets
- Update w fields
Superconductivity (CINECA)

- Partner: CINECA
- TurboRVB
- Quantum Monte Carlo simulations to study superconducting mechanisms in high critical temperature materials
- Fortran 90
Superconductivity (CINECA)

- Phases:

  - Initialisation
  - Scratch
  - Uptable
  - Upgreen
  - Branching
  - Finalisation

- Main loop
  - MC Walker loop
• TAU communication matrix (number of calls)
  – Mainly nearest neighbours communication
Superconductivity (CINECA)
Conclusions

• DEEP provides a novel and flexible platform

• The programming model uses well known APIs
  – Plus a few changes to leverage asynchronicity and easy offloading

• Applications will be ported to it
  – Division between Cluster and Booster is clear now
Thank you