Demo: Programming efficiently
HPC apps

Richard Kavanagh (ULEEDS)
HiPEAC Demo, 26th October 2017
TANGO Toolbox

Valuable Tools

Simplify & Optimize
Heterogeneity

abstract & control

- Programming Model
- ALDE
- Runtime
- Device Supervisor
- Energy Modeler
- Monitoring Infrastructure
- Self-Adaptation Manager

Simplify

Optimize
Global Vision of the DEMO

- HPC Developer
- The developer wants to create an application for:
  - CPU
  - GPU
  - CPU+GPU
- The developer is also interested into adapt the behaviour of the application depending on the conditions
Workflow

HPC Developer

TANGO TOOLBOX

Programming Model

IDE

Developing
Goal
- Easy programming of applications and efficient execution in distributed heterogeneous computing environment

Demo:
- Show how the same code can be adapted to the available infrastructure
Block Matrix multiplication

- Matrix is divided in NxN blocks of MxM size.

- Code

```java
// Main Code
for (int i = 0; i < N; i++){
    for (int j = 0; j < MSIZE; j++){
        for (int k = 0; k < MSIZE; k++){
            multiplyBlock(A[i][k], B[k][j], C[i][j]);
        }
    }
}

// Task Code
public static void multiplyBlock ( Block a, Block b, Block c )
{
    for( int i = 0; i < M; i++ )
        for( int j = 0; j < M; j++ )
            for ( int k = 0; k < M; k++ )
                c.data[i][j] += a.data[i][k] * b.data[k][j];
}
```

- Example

- Each block multiplication is parallelized by OmpSs

```
A
A_{0,0}  A_{0,1}

B
B_{0,0}  B_{1,0}

C
C_{0,0}

A_{0,0}  B_{0,0} + A_{0,1}  B_{1,0} = C_{0,0}
```
Matmul Task Definition

interface Matmul
{
    @Constraints(processors={ @Processor(ProcessorType=CPU, ComputingUnits=5)});
    void multiplyBlock ( in Block block1, in Block block2, inout Block Block3, in int M);

    @Constraints(processors={@Processor( ProcessorType=GPU, ComputingUnits=1)});
    @Implements(multiplyBlock);
    void multiplyBlock_GPU ( in Block block1, in Block block2, inout Block Block3, in int M);
};

void multiplyBlock (Block block1, Block block2, Block block3, int M) {
    for (int i=0; i<M; i++) {
        for (int j=0; j<M; j++) {
            #pragma omp task inout (block3.data[i][j]) in (block1.data[i][0..M-1], block2.data[0..M-1][j])
            for (int k=0; k<M; k++) {
                block3.data[i][j] += block1.data[i][k] * block2.data[k][j];
            }
        }
    }  #pragma omp taskwait
}

//GPU
void multiplyBlock_GPU (Block block1, Block block2, Block block3, int M) {
    Muld(block1.data, block2.data, block3.data, M);
}

//Kernel.h
#pragma omp target device(cuda) copy_deps ndrange(2, 64, 64, 32, 32)
#pragma omp task in(A[0:NB*NB], B[0:NB*NB]) inout(C[0:NB*NB])
__global__ void Muld(double* A, double* B, int wA, int wB, double* C, int NB);
ALDE

Source code from Programming Model

Execution Configuration 1: CPUs

Execution Configuration 2: GPUs

Execution Configuration 3: CPUs + GPUs

Creates a Singularity Container

Uploads Container to Device Supervisor
Workflow

HPC Developer → IDE

Programming Model → ALDE

App src → Deploying

App Binaries

Device Supervisor

Deploying
Device Supervisor

- It is the software responsible for the resource management and allocation of jobs.
  - It matches the jobs needs for compute power to the resources characteristics
  - It maintains queues of jobs and provides scheduling based on particular constraints and algorithms
- It can supervise several nodes that contain heterogeneous devices
- It follows the job execution and can provide accounting and profiling results once the job is finished
- It offers various APIs that can be used by the ALDE.
- It is represented by the open-source RJMS Slurm.
Workflow

TANGO TOOLBOX

HPC Developer

IDE

Heterogeneous Devices

App Scheduling

Device Supervisor

Monitor Infrastructure

App Binaries

Running

TANGO TOOLBOX
Monitor Infrastructure

- It monitors the heterogeneous resources
- Specially in terms of energy usage, utilization and temperature
- It relies on Collectd and a set of custom plugins (for NVIDIA GPUs and Xeon Phi coprocessors)
- It uses InfluxDB (historic values) and Grafana (GUI)
- It offers a REST API to connect to other TANGO components
  - Self Adaptation Manager, Energy Modeller, Runtime
Adapting

Programming Model

ALDE

Device Supervisor

App Binaries

src

IDE

HPC Developer

TANGO TOOLBOX

Self-Adaptation Manager

Monitor Infrastructure

App Scheduling

Heterogeneous Devices

Energy Modeler
Energy Modeller

Role and Purpose:
- Maps utilisation of resources to power consumption
- Determines application based power consumption
- Allows forecasting of power consumption
- Aids adaptation by understanding power consumption of an application

Grafana Dashboard: localhost:3000/dashboard/db/application-power
Energy Model Calibration

![Energy Model Calibration Diagram]

- **Power (W)** vs **Time (s)**
- **ns51** graph showing the relationship between **Power (W)** and **Utilisation (%)**
- Calibration Point: 
  - **CPU Utilisation**: \(<\text{value, time}>\)
  - **Measured Power**: \(<\text{value, time}>\)
  - **Index, CPU Utilisation, Measured Power**

Mathematical model:

\[ y = 1.0921x + 122.25 \]

\[ R^2 = 0.9934 \]
1. When to adapt?  
   – **Time**: reactive, proactive

2. Why do we have to adapt?  
   – **Reason**: context, resources

3. Where do we have to implement change?  
   – **Level**: application, software, resource

4. What kind of change is needed?  
   – **Technique**: parameter, structure

5. How is the adaptation performed?  
   – **Control**: approach, decision criteria
Self-Adaptation Manager

- It monitors values from CollectD and SLURM
- It has rule sets that allows it to decide when to adapt

It can detect events such as:

• Boundary conditions on measurements
• Idle hosts
• Idle hosts with accelerators
• Application starting / completion
• Applications approaching their deadline
• Events driven by cron expressions
The SAM generates QoS based events such as:

- idle hosts + idle hosts with accelerators,
- application starting / completion and applications approaching their deadline.
- cron statements in the format (Unique Id, Agreement Term, Cron Statement)

<table>
<thead>
<tr>
<th>Unique Id</th>
<th>Agreement Term</th>
<th>Comparator</th>
<th>Event Type</th>
<th>Guarantee Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IDLE_HOST</td>
<td>EQ</td>
<td>WARNING</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>APP_FINISHED</td>
<td>EQ</td>
<td>WARNING</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>CLOSE_TO_DEADLINE</td>
<td>EQ</td>
<td>WARNING</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>CPUSum</td>
<td>EQ</td>
<td>WARNING</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>power</td>
<td>GT</td>
<td>SLA_BREACH</td>
<td>400</td>
</tr>
<tr>
<td>6</td>
<td>cpu-measured</td>
<td>GT</td>
<td>WARNING</td>
<td>50</td>
</tr>
<tr>
<td>7</td>
<td>HOST:ns50:cpu-measured</td>
<td>GT</td>
<td>WARNING</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>app_power:COMPSs:*</td>
<td>GT</td>
<td>SLA_BREACH</td>
<td>300</td>
</tr>
<tr>
<td>9</td>
<td>app_power:RK-Bench:*</td>
<td>GT</td>
<td>SLA_BREACH</td>
<td>50</td>
</tr>
</tbody>
</table>
The SAM uses mapping rules to go from events to response type.

<table>
<thead>
<tr>
<th>Agreement Term</th>
<th>Direction</th>
<th>Response Type</th>
<th>Event Type</th>
<th>Lower bound</th>
<th>Upper bound</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISABLED_CPUTot</td>
<td>EQ</td>
<td>PAUSE_APP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DISABLED_IDLE_HOST+ACCELERATED</td>
<td>EQ</td>
<td>UNPAUSE_APP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DISABLED_cpu-measured</td>
<td>GT</td>
<td>PAUSE_APP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DISABLED_HOST:ns50:cpu-measured</td>
<td>GT</td>
<td>PAUSE_APP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APP_FINISHED</td>
<td>EQ</td>
<td>KILL_SIMILAR_APPS WARNING</td>
<td>0</td>
<td></td>
<td></td>
<td>application=RK-Bench</td>
</tr>
<tr>
<td>APP_FINISHED</td>
<td>EQ</td>
<td>KILL_SIMILAR_APP S WARNING</td>
<td>0</td>
<td></td>
<td></td>
<td>application=COMP</td>
</tr>
<tr>
<td>app_power:RK-Bench</td>
<td>GT</td>
<td>HARD_KILL_APP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>app_power:RK-Bench:*</td>
<td>GT</td>
<td>HARD_KILL_APP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Resulting in adaptations occurring:

<table>
<thead>
<tr>
<th>Time</th>
<th>Application ID</th>
<th>Deployment ID</th>
<th>Host</th>
<th>Action Type</th>
<th>Agreement Term</th>
<th>Guaranteed Value</th>
<th>Raw Value</th>
<th>Guarantee Operator</th>
<th>Able to Respond</th>
<th>Action Performed</th>
</tr>
</thead>
<tbody>
<tr>
<td>150521</td>
<td>KILL_SIM</td>
<td>ILAR_APP APP_FINISH</td>
<td>S</td>
<td>ED</td>
<td>0</td>
<td>0</td>
<td>EQ</td>
<td>TRUE</td>
<td>TRUE</td>
<td></td>
</tr>
<tr>
<td>5564</td>
<td>COMPSs</td>
<td>3162</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Rule Flexibility

Sample Cases for Adaptation:
- Case 1 - Hardware Becomes Available (QoS Improvement)
- Case 2 - Power Smoothing, energy saving
- Case 3 - Benchmarking and search for fastest/most efficient application

Forms of Adaptation:
- Change wall time
- Kill application
- Kill similar applications
- Pause/resume application
- Pause/resume similar applications
- Reselect accelerators
- Add/remove CPUs from jobs (pending jobs only)
- Add/remove nodes from jobs (pending jobs only)
- Oversubscribe/Exclusive access to resources (pending jobs only)
Triggering the reversing of a rule after a period of time:

<table>
<thead>
<tr>
<th>Agreement Term</th>
<th>Direction</th>
<th>Response Type</th>
<th>Event Type</th>
<th>Lower bound</th>
<th>Upper bound</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISABLED_CPUTot</td>
<td>EQ</td>
<td>PAUSE_APP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DISABLED_IDLE_HOST+ACCELERATED</td>
<td>EQ</td>
<td>UNPAUSE_APP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>app_power:RK-Bench:*</td>
<td>GT</td>
<td>PAUSE_APP</td>
<td>SLA_BR</td>
<td>-</td>
<td>10000</td>
<td>10000 application=RK-Bench;UNPAUSE=10</td>
</tr>
<tr>
<td>!app_power:RK-Bench:*</td>
<td>EQ</td>
<td>UNPAUSE_APP</td>
<td>WARNING</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APP_FINISHED</td>
<td>EQ</td>
<td>KILL_SIMILAR_APP</td>
<td>WARNING</td>
<td>0</td>
<td>0</td>
<td>application=RK-Bench</td>
</tr>
<tr>
<td>APP_FINISHED</td>
<td>EQ</td>
<td>KILL_SIMILAR_APP</td>
<td>WARNING</td>
<td>0</td>
<td>0</td>
<td>application=COMPSs</td>
</tr>
<tr>
<td>app_power:RK-Bench</td>
<td>GT</td>
<td>HARD_KILL_APP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>app_power:RK-Bench:*</td>
<td>GT</td>
<td>HARD_KILL_APP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Temporal Parameters:

START_TIME=9:00;END_TIME=17:00;DAY_OF_WEEK=1111100

START_TIME=9:00 – Indicating rule can only fire after 9am
END_TIME=17:00 – Indicating rule can only fire until 5pm.
DAY_OF_WEEK=1111100 – Indicating Rule fires Monday-Friday
TANGO Toolbox
Valuable Tools

abstract & control
Simplify & Optimize
Heterogeneity

Simplify
Optimize

Programming Model
ALDE
Runtime

Device Supervisor
Energy Modeler
Monitoring Infrastructure
Self-Adaptation Manager

Simplify
Optimize
Thank you!