The BarbequeRTRM Framework

*Run-Time Resource Management on Many-Core STHORM Platform*

Speaker: **Patrick Bellasi**
bellasi@elet.polimi.it
Outline

- Current SThorn Run-Time components
- Overview of BarbequeRTRM Framework
- SThorn Platform Integration in BarbequeRTRM
- Applications Integration Examples
- Conclusions
Run-Time Resources Management (RTRM) is about finding the optimal trade-off between QoS requirements and resources availability.

- **Target scenario**
  - Shared HW resources
    - many-core devices, e.g. SThorm platform, are complex systems
    - process variations and run-time issues, e.g. thermal, aging, etc.
  - Mixed SW workloads
    - resources sharing and competition
      - among applications with different priorities and time-varying requirements

- **Simple solutions are required**
  - support for frequent change of use-case context
  - suitable for both critical and best-effort applications
Introduction
The SThorm Platform Run-Time (1/2)

- Two main software modules
  P2012 **Device Driver** (DD)
  
  *initialize the Host-Fabric (H-F) shared memory to support configuration and communications, via shared data structures*
  
  *manage interrupts generated on the Fabric side*
  
  *deploy and start the Fabric “primordial run-time” (p12Runtime.exe)*
  
  *setup communication channels*

  P2012 **Library** (libp2012.so)
  
  *device driver wrapper for the applications support APIs*
  
  *queue, buffers and messages management*

- H-F communication through
  
  *messages on shared communication queues*
  
  *used to define a control channel*
  
  *dedicated buffers*
  
  *used to exchange big amount of data*
Introduction
The SThorm Platform Run-Time (2/2)

- Additional software services
  
  P2012 **Daemon** (p2012d)
  
  *has exclusive access to DD control functions*
  *i.e. all the calls but “send message”*

  **OpenCL Daemon** (oclDaemon)
  
  *Host side component, required to run OpenCL applications*

  Resources and Power Management
  
  *simple interface to query Fabric status or set clusters frequencies*

- The p2012d provides a pretty simple and limited support to resources management
  expected to be extended along the roadmap
Main issues

How to **co-schedule** OCL and NPM applications?
How to co-schedule multiple NPM applications?
How to give **priority** to some workloads?

Missing support for **system-wide** policies
Power/Thermal
Energy/QoS

Need for a more advanced Run-Time Resource Manager
The BarbequeRTRM
Overall View on Run-Time Resource Management

BarbequeRTRM on SThorm Platform

The BarbequeRTRM
Overall View on Run-Time Resource Management

We cannot cover internal details
please check project website and past presentations

Bellasi and Massari, Tutorial - “The BarbequeRTRM Framework
2PARMA Framework for Run Time Resource Management
of Multi-Core Computing Platforms”. Fall School Forest,
Freudenstadt, 09/2012.

Complete Framework Review
+ Hands On Sessions

Results on Multi-Core NUMA machine

Bellasi et.al., ”A RTRM proposal for multi/many-core platforms and
reconfigurable applications”. ReCoSoC 2012.

Official Project Website

http://bosp.dei.polimi.it
The BarbequeRTRM
What's Next?

System-Wide RTRM
Coarse grained control on platform available resources:
- resource accounting, partitioning and abstraction
- high-level HW events handling e.g., critical conditions, faults...
- manage applications priorities
- power/thermal “coarse tuning”

Application-Specific RTM
Fine grained control on application allocated resources:
- task ordering
- virtual processor assignment
- DVFS
- application parameters monitoring

Legend
X SW Interface (API)
Y SW/HW Meta-data

BarbequeRTRM on SThorm Platform
**Application Integration Layer**

---

**BarbequeRTRM on SThorm Platform**

**Dynamic Code Generation**
- Critical Apps
- Best-Effort Apps
- RTLib
- Res Accounting
- Res Partitioning
- Res Abstraction
- MRAPI
- Platform Proxy
- Platform Driver
- Platform Firmware

**Application-Specific RTM**
-Fine grained control on application allocated resources:
  - task ordering
  - virtual processor assignment
  - DVFS
  - application parameters monitoring

**System-Wide RTRM**
-Coarse grained control on platform available resources:
  - resource accounting, partitioning and abstraction
  - high-level HW events handling e.g., critical conditions, faults...
  - manage applications priorities
  - power/thermal “coarse tuning”

**Legend**
- SW Interface (API)
- SW/HW Meta-data

---

**Politecnico di Milano**
Goal: smooth transition to BarbequeRTRM

the BBQ daemon is just another p2012d client

apps are interfaced via a provided RTLib library

fabric run-times are extended with resources control functions (C)

a shared struct (MDD) in FC memory interfaces BBQ with run-time control functions (C)
Platform Integration Layer
The Managed Device Descriptor (MDD)

- A pre-defined data structure in FC memory mapped at beginning of the physical address
- Defines a set of RTRM related “descriptors”

- Platform identification and run-time capabilities
- Configuration of F2H events notifications
- Array of Clusters and PEs descriptors
- Array of EXCs Constraints

Memory requirements: ~5KB
<17% of 32KB FC TCDM, for a platform with 4 Cluster, 16 PEs and up to 64 concurrent EXCs
Platform Integration Layer
The Platform Descriptor

- Parsed by BBQ to get info on available resources
- Three main components
  - **Cluster** descriptor
    - thermal sensors status and configurable thresholds
    - deployed run-time, i.e. OCL or NPM
    - power model, i.e. \(\{\text{min, max, cur}\}, \{\text{Hz, W}\}\)
  - DMA descriptors
  - PEs descriptors
  - **DMA** descriptor
    - bandwidth allocation policy
      - i.e. \(\{\text{max, cur}\ \text{bandwidth}\}\)
  - **PE** descriptor
    - power model
      - i.e. \(\{\text{min, max, cur}\}, \{\text{W}\}\)
- Array of **run-time constraints sets (RCS)**
  asserted by the BarbequeRTRM daemon
  one for each active **F-side EXecution Context (F-EXC)**
  => up to 64 RCS

- Each constraints set
  unique associated to a **H-side EXecution Context (H-EXC)**
  defines a set of run-time engine specific constraints
  host run-time controller specific data-structures

  **MTE (OpenCL)**
  average amount of F time

  **RTE (NPM)**
  cluster bitmap

**Features under development**
... but something is already working!
OpenCL applications schedule WorkGroups (WG) which cannot be preempted while executing on F-side. Kernels co-scheduling is not convenient, e.g., scheduling overheads, limited TCDM, no memory protection.

Solution: WGs are “properly dequeued.” Computation time on fabric side is accounted for each EXC. A “desirable” fabric time quota is asserted by BBQ for each active EXC, using a dedicated constraints descriptor. => run-time dequeue WGs trying to enforce this time quota.

Best-effort approach, precision depends on: WGs average execution time, i.e., the lower the better. Time frame considered for time quota computation.
Platform Integration Layer
The NPM Run-Time Integration

- NPM applications deploy tasks on the fabric which are directly mapped on Clusters and PEs
  *i.e. tasks allocation is defined by the application*

- Solution: Clusters and PEs are “properly accessed”
  applications ask resident run-time for Clusters IDs
  *e.g. I need a cluster, which one could be used?*
  the set of **authorized clusters** is identified by BBQ
  *for each active EXC, using a dedicated constraints descriptor*
  => run-time allows kernel deployment based on this decision

- Requires a modification of the NPM API
  applications needs to query for resources IDs
  *i.e. probably just small code re-arrangements*
Platform Integration Layer

BarbequeRTRM on SThorm Platform

Application-Specific RTM
Fine grained control on application allocated resources:
- task ordering
- virtual processor assignment
- DVFS
- application parameters monitoring

System-Wide RTRM
Coarse grained control on platform available resources:
- resource accounting, partitioning and abstraction
- high-level HW events handling e.g., critical conditions, faults...
- manage applications priorities
- power/thermal “coarse tuning”

Legend
- X SW Interface (API)
- Y SW/HW Meta-data

BarbequeRTRM

Diagram:
- Critical Apps
- Best-Effort Apps
- Res Accounting
- Res Partitioning
- Res Abstraction
- Platform Driver
- Platform Firmware
- Task Mapping
- DDM
- Dynamic Code Generation
- Res Accounting
- Res Partitioning
- Res Abstraction

BarbequeRTRM on SThorm Platform

POLITECNICO DI MILANO
Run-time **reconfigurable** workloads

*Example:* Scalable Video Coding (SVC)

- Single input stream, different decoding configurations
- Different decoding profiles which correspond to different quality-vs-performances requirements
Application Integration Layer
Target applications (2/2)

- Stream processing applications
  which means not only multimedia processing
  \textit{e.g.} packet sniffing and analysis, pattern matching, ...

- Well defined \textbf{Abstract Execution Model} (AEM)
  loop of actions, until no more workload to process
  \textit{Setup, Configure, Running, Monitor}
Application Integration Layer
Run-Time Library (RTLlib)

- Abstracts the communication channel using “threaded FIFOs”, planned Binder/DBus support
- Hides Synchronization-Protocol details required to sync reconfiguration among multiple applications
- Provides an AEM-based abstract API
- Provides integrated support for application profiling
AEM Abstract API

- **callbacks** based with default implementations

- hide all the RTM boilerplate code
Activity

interacts with service via **function calls**
  e.g. start/stop processing, etc.
receive notifications via **async messages**
  e.g. progress, configuration switch, etc.

Custom Service

extends a **BbqueService**
  which implements **native function calls**
implements **RTLib AEM callbacks**
provides message dispatching

BarbequeRTRM

provides a JNI wrapper for the **RTLib**
supporting bi-directional calls
manages resources on **many-core** platforms

---

Examples of Application Integration
Integration of “interesting/real” Applications (1/2)

- Partial integration of PARSEC v2.1 benchmarks
  Bodytrack and Ferret (pthreads version) already done
  *two different examples: data-parallelism vs pipelining*

- Example of the required effort:
  code re-arrangement to support run-time re-configurations

RTLib integration (mostly copy-and-paste code)

Because this application is **not designed** for Run-Time management
Examples of Application Integration
Integration of “interesting/real” Applications (2/2)

- **2PARMA reference workloads**
  Multiview, OpenCL on x86, OpenCL-P2012 WIP
  Scalable Video Coding (SVC), both x86 and NPM-P2012

- **Internal developed applications**
  OCVDemo, OpenCV+TBB on x86
  http://youtu.be/4DlPqY8F6SY

- **Overall considerations**
  most of the effort spent on developing run-time tunable applications
  or adapting applications not designed to be run-time tunable
  RTLib integration is straight-forward
  almost only copy-and-past code
### Examples of Application Integration

**Android Workloads – FaceDetect (by STMicroelectronics)**

- **Re-engineered just to integrate with BarbequeRTRM and exploit the Android Camera**

<table>
<thead>
<tr>
<th>Activity</th>
<th>Camera</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>startCameraPreview</td>
<td>PreviewCallbacks</td>
<td>OnRun Callback</td>
</tr>
<tr>
<td>Show preview on surface</td>
<td>MSG_FRAME [start]</td>
<td>BarbequeRTRM</td>
</tr>
<tr>
<td>Draw rectangles</td>
<td>getFrame</td>
<td>STHORM</td>
</tr>
<tr>
<td></td>
<td>MSG_FRAME [end, coords]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Return coordinates</td>
</tr>
</tbody>
</table>

**Native FDetectRun**

**YouTube Screencast:** “FaceDetect on BarbequeRTRM running on Android”

**GIT Repo:** [http://goo.gl/RE3t3](http://goo.gl/RE3t3)
Conclusions
The Barbeque OpenSource Project (BOSP)

- Based on (a customization of) Android building system freely available for download and (automatized) building

Framework dependencies
External libs, tools, ...

Framework Sources
BarbequeRTRM, RTLlib

Framework Tools
PyGrill (loggrapher), ...

Contributions
Tutorials, demo

Public GIT repository
https://bitbucket.org/bosp
Thanks for your attention!

If you are interested, please check the project website for further information and keep update with the developments

http://bosp.dei.polimi.it