BBQ - The Barbeque RTRM
2PARMA Framework for Run Time Resource Management for Multi-Core Computing Platforms

overall view on tool goals, requirements and design

Speaker: Prof. William Fornaciari
fornacia@elet.polimi.it

Chief developer: Dr Patrick Bellasi
bellasi@elet.polimi.it

Politecnico di Milano
Dipartimento di Elettronica e Informazione
Via Ponzio 34/5 – 20133 Milano, Italy

Last review: Jan, 19 2012
Summary

- Goals and requirements of RTRM of resources
- BBQ in 2PARMA
- BBQ architecture
- Features of the V.0.6 (Angus)
- 1 minute demo (video)
- Genesis of the BBQ name
... living in this world is hard for the designers...
Find the optimal **trade-off** between QoS requirements and resources availability

- Shared HW resources, mixed SW workloads
  - upcoming many-core devices are complex systems
  - process variations and run-time issues
  - resource sharing and competition
- Simple SW solutions are required
  - support frequently changing use-case
  - suitable for both critical and best-effort applications
**The RTRM Tool**

**Objectives of RunTime Management**

- Multiple devices, subsystems
  Heterogeneous -> Homogeneous (Many-Cores)

  *Scalability and Retargetability*

- Shared resources among different devices and applications
  Computation, memory, energy, bandwidth…

  *System-wide resources management*

- Multiple applications and usage scenarios
  Run-time changing requirements

  *Time adaptability*
Support monitoring, management and control at different granularity levels to reduce overheads

Different granularity

- accelerated application
- operating system
- computation fabric
- computation clusters

How to reduce the control complexity?

Each granularity level collects requirements from higher level and it provides constraints to lower levels
Map "virtual resources" on "physical resources" at run-time to achieve optimal platform usage

Considering run-time phenomena
- process variation
- hot-spot and failures
- workload variation

How to support optimal system resource exploitation?
Virtual resources representation to support accounting; map on physical ones at run-time to handle variations
Grant resources to **critical workloads** while optimize resource usage by **best-effort workloads**

Considering a **mixed-workload** scenario
- critical workloads could be off-line optimized (e.g., using DSE)
- other workloads runs concurrently

How to handle resources granted to **critical applications**?

**Dynamically grant these resources to best-effort workloads while not required by critical ones**
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC.1 monitor resource performance</td>
<td>Export an updated view of the usage and behaviors of each subsystem, and their resources, with different details levels.</td>
</tr>
<tr>
<td>RC.2 dynamic resource partitioning</td>
<td>Grant resources to critical workloads while dynamically yield these resources to best-effort workloads when (and only while) they are not required by critical ones, thus optimize resource usage and fairness.</td>
</tr>
<tr>
<td>RC.3 resource abstraction</td>
<td>Handle a decoupled resource view between the users and the underlying hardware.</td>
</tr>
<tr>
<td>RC.4 multi-objective optimization policy</td>
<td>System-wide and multiple metrics optimization policy.</td>
</tr>
<tr>
<td>RC.5 dynamic optimization policy</td>
<td>Support multiple and tuneble policies to fit well specific optimization scenarios.</td>
</tr>
<tr>
<td>RC.6 low run-time overheads</td>
<td>Reduced impact on the performances of the controlled system.</td>
</tr>
</tbody>
</table>
The RTRM Tool
Links With Other 2PARMA Work Packages
The Barbeque Approach to RTRM

*an overall view on proposed tool architecture*
The Barbeque RTRM
Overall architectural view

Application-Specific RTRM
- Fine grained control on application allocated resources:
  - task scheduling
  - (indirect) power "fine tuning"

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

Barbeque provided code/modules

Legend
- SW Interface (API)
- SW/HW Meta-data
Resource Partitioning

Formal model and heuristic for resources partitioning and optimization
A formal model has been introduced - based on ILP

allows to identify the optimal resource partitioning among all active application workloads

- considering all the main aspects of modern platforms and applications
  i.e. clustered resources (e.g., PEs), mixed-workload, reconfiguration overheads, applications working modes and QoE expectations, power management and optimization

An run-time efficient heuristic has been designed

- prototype implementation (just for numerical evaluation)
- very low performances-degradation with respect to the formal model (which is not run-time computable)
- on going integration within the main RTRM framework
Resource Partitioning
Tunable resource partitioning policy (example results)

High-Performance (HeuHP) vs Low Performance (HeuLP)
resource partitioning

Realistic usage scenario:
- 20 applications: arriving at different times, different permanence time, different priorities

Worst Case Analysis

Low Performance Policy
- heuristic partitioning error increase because of the enforced aggressive resources preservation
- this turns out on rejecting more applications
- ... but also reduced resources

High Performance Policy
- heuristic partitioning always <22% wrt the value of the optimal one
Barbeque RTRM
v0.6 (Angus) – Last stable release
Barbeque RTRM
Framework Architecture - v0.6 (Angus)

- Complete support for application development
  RTLib
    * Plain and AEM-based API
  Applications mgmt
  Resource accounting
  Scheduling policies
    * randon (stress test)
  YaMCA
  Synchronization Policy
  SASB
- No platform integration
  WIP c-group based platform proxy
- Complete description of application requirements
  System priority, Application Working Modes (AWMs), Resources requirements (platform specific)
- Deployed at application installation time into global system folder
- XML Schema still to be freezeed possibly merge with Android manifest?
Based on (a customization of) Android building system freely available for download and (automatized) building

Framework dependencies
External libs, tools, ...

Framework Sources
BarbequeRTRM, RTLib

Framework Tools
PyGrill (loggrapher), ...

Contributions
Tutorials, demo
Pre-compiled and configured framework running on a VirtualBox machine
Grant resources to **critical workloads** while optimize resource usage by **best-effort workloads**

- Considering a mixed-workload scenario
  
  critical workload (High Priority)
  
  **Application:** SVCDemo
  
  app using SVC Video Decoding library by HHI (x86_32 version)
  
  other workloads runs concurrently (Lower Priority)
  
  **Applications:** 1 SVCDemo + 3 BbqRTLibTestApp
  
  **BbqRTLibTestApp** as background “disturbing” apps

- **Goals:** **application integration** and **dynamic resource partitioning**
  
  according to apps priority and resources demand
  
  synthetic recipes (no DSE based), mechanisms evaluation (behaviors, overheads, stability,...)

  **no policies nor platform control assessment**
Barbeque RTRM
Framework Demo on Dynamic Resource Partitioning

Resources Scheduling

Barbeque RTRM

Live demo available...
Because of its “sweet analogy” with something everyone know...

### QoS
how good is the grill

### Applications
the stuff to cook

### Overheads
Cook fast and light

### Task mapping
the chef’s secret

### Reliability Issues
dropping the flesh

### Priority
how thick is the meat or how much you are hungry

### Mixed Workload
sausages, steaks, chops and vegetables

### Thermal Issues
burning the flesh

### Policy
the cooking recipe

### Resources
coals and grill

BBQ: the Barbeque RTRM
Barbeque RTRM
v0.8 (Bacon) – Next stable release (02/2012)
- RTLib instrumentation for “perf” support
  standard Linux interface to access performance counters
- Allows AWM-specific monitoring
  “for-free”, for each application integrated with the RTLib
  could support integration with DSE techniques
  negligible run-time overheads
  could support BBQ run-time monitoring and AWM tuning

- Simple usage by exec environment configuration
  `BBQUE_RTLIB_OPTS="p" <your_application>`

- Many flags to customize the reporting
  detail level (simple, +D$, +I$, +Bus cycles)
  kernel overheads
  RTLib overheads
  [output format (text vs CSV)]
Example: Bodytrack with 8 or 4 threads
- Started the 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

Because this application is **not designed** for Run-Time management

RTLib integration (mostly copy-and-paste code)
Integration with Kconfig
simpler configuration of BBQ compilation/installation options
Initial support for Android development

ARM cross-toolchain generation

*based on Linaro patches*

update.zip generation and deployment (via ADB)

*both on SDK emulator and physical device*

to run both BBQ and RTLib native applications

Now working on access BBQ from JVM applications

either by “wrapping” RTLib into an Android Service

… or developing a binder plugin for BBQ
Example, BBQ & TestApp running on Android emulator
Barbeque RTRM
Platform Integration

- Initial support for platform integration
  generic Linux SMP machine
  portable solution, based on Linux Control Group
  \textit{CPU and Memory nodes assignments}
  \textit{CPU quota (work-in-progress, requires kernel 3.2)}

- Support both resources monitoring and control
  pre-configured CGroup to define BBQ controlled resources
  \textit{at Barbeque start, by parsing a pre-defined cgroup}
  \textit{… than Barbeque takes control over these resources}
  tun-time generation of new CGroup to control applications
  \textit{requires RTLib integration (of course)}

- Similar solution could fit for P2012 integration
  requires just a new ad-hoc CGroup subsytem
  support complete “decoupling” among BBQ and P2012
Working on the definition of a new “modular policy” replacing by improving the current on (YaMCA)

Multi-objective optimization goals:

Partition available resources ($R$) among active applications ($A$), considering:

- $A$ properties: priority and resource requirements
- $A$ status: current selected AWM, current assigned resources
- $R$ properties: power/thermal status
- Overheads: migration/reconfiguration costs
- Power budget: clusters switch-on costs

Increase overall system value considering discrete values changes

Stability considering a thread-off between reconfigurations and benefits

Robustness absorbing small modification on resources requirements/availabilities
The new scheduler policy is designed to enforce stability and robustness:

- **promoting** the scheduling of AWM which allows to improve optimization goals
- **demoting** the scheduling of AWM which degrade solution metrics such as stability and robustness
Because of its “sweet analogy” with something everyone know...

QoS
how good is the grill

Applications
the stuff to cook

Overheads
Cook fast and light

Priority
how thick is the meat
or
how much you are hungry

Task mapping
the chef's secret

Mixed Workload
sausages, steaks, chops
and vegetables

Reliability Issues
dropping the flesh

Thermal Issues
burning the flesh

Policy
the cooking recipe

Resources
coals and grill

Task mapping
the chef's secret