The BarbequeRTRM Framework

Targeting Applications and Platform "Variability" Challenges

Prof. William Fornaciari
william.fornaciari@polimi.it

Politecnico di Milano - DEIB
Dipartimento di Elettronica, Informazione e Bioingegneria
Milano, Italy

POLITECNICO MILANO

2 PARMA

SEVENTH FRAMEWORK PROGRAMME

Platforms Evolution

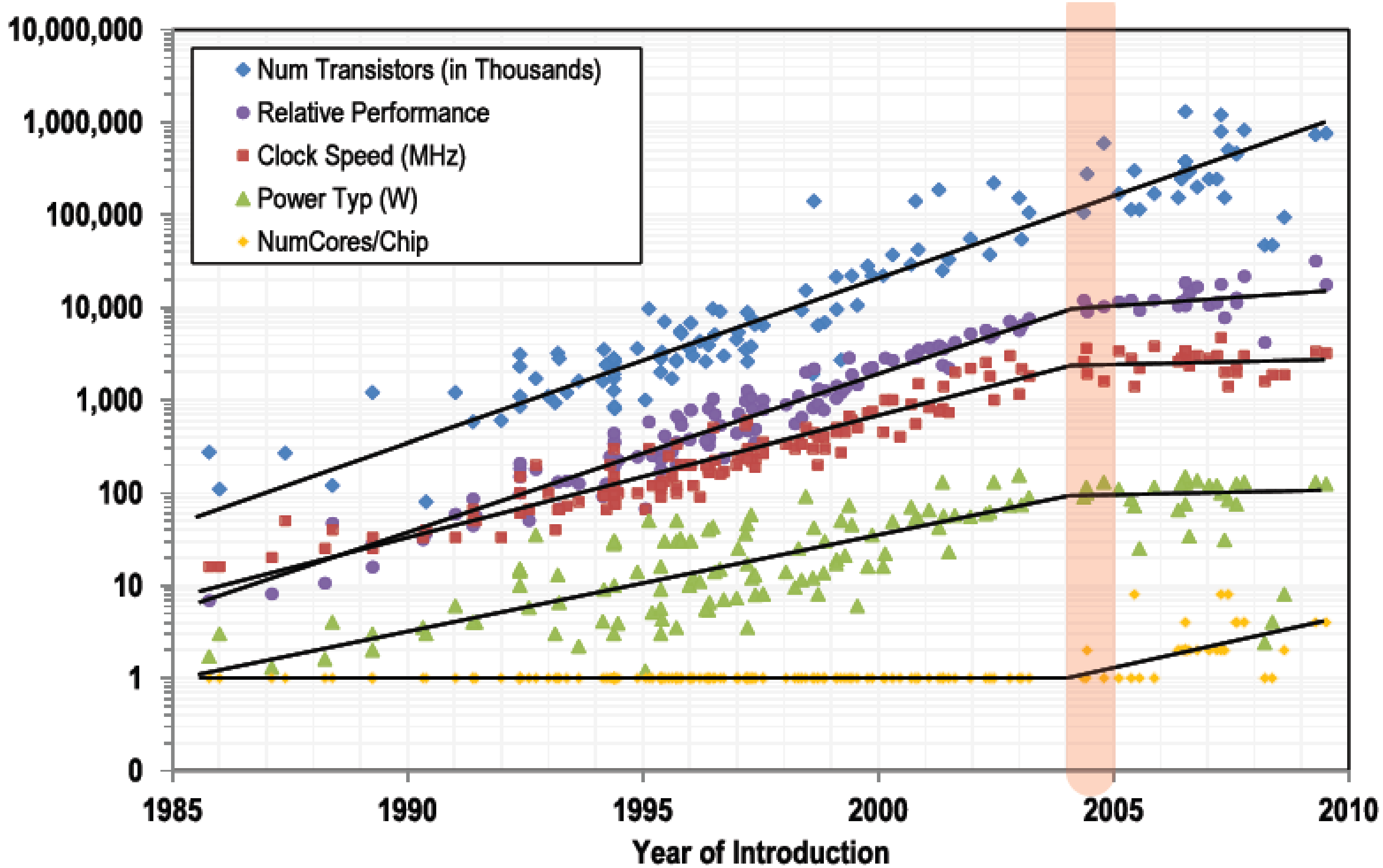
Some big: good... many small: better!

Outline of the keynote

- Introduction: towards *multi-problem* and multi-core
  - Challenges for new generation of applications
- Effective and flexible exploitation of new platform capabilities
  - Adaptability
- The BBQ RunTime Resource Management approach
  - Tradeoff, achievements, the BOSP open source project
- Screen-cast of use cases
  - BBQ in action
- Work in progress
  - Roadmap and new FP7 projects

Platforms Evolution
Power Trend: The 2004 Inflection Point

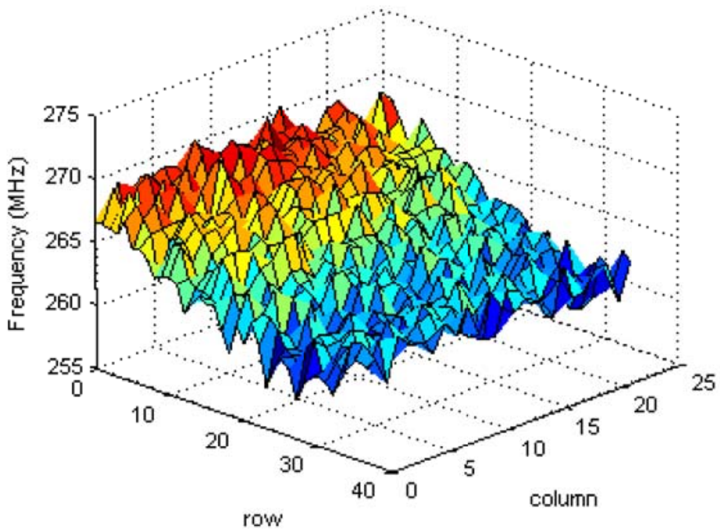
- From single-core to multi-core processors



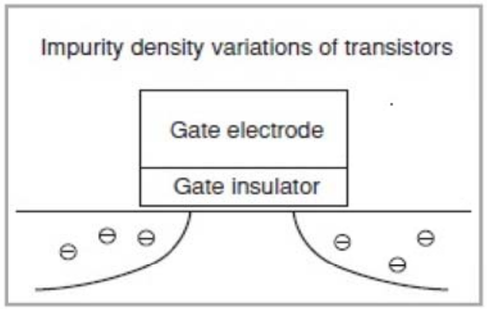
Source: **B. Falsafi. "Reliability in the Dark Silicon Era". IOLTS2011 Keynote, July 2011.**

Platforms Evolution
Power Trend: Process Variation

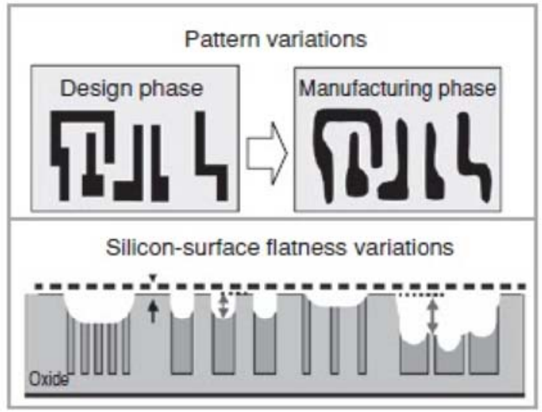
- STHORM/P2012 69 multi-core 28nm SoC, need to:
  - Consider frequency control at SoC and cluster granularity
  - Introduce PVT (Process, Voltage, Temperature) sensors
  - Joint design of firmware and OS layers



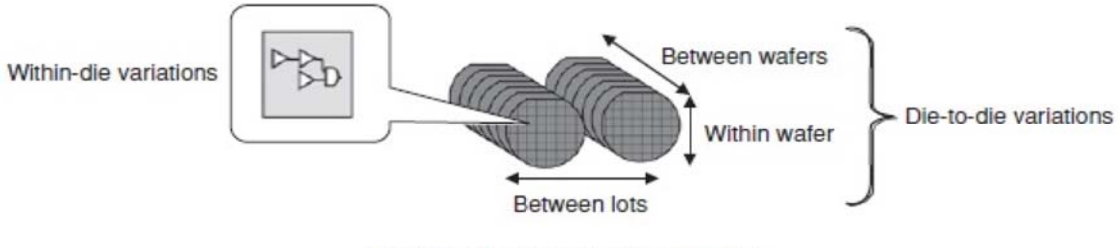
***Source*: Dr Pete Sedcole - Altera Cyclone II EP2C35.**



(a) Random variations



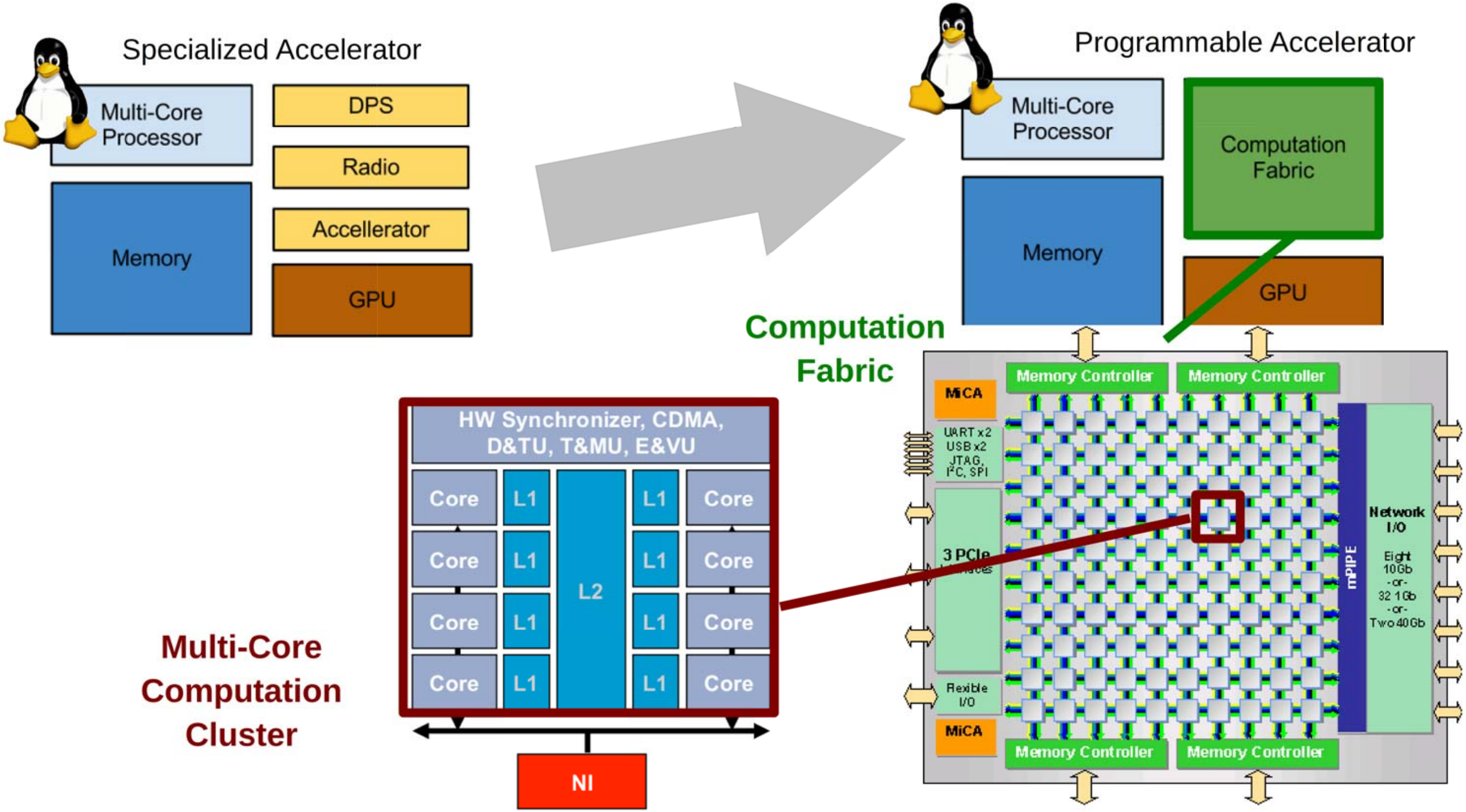
(b) Systematic variations



(c) Within-die and die-to-die variations

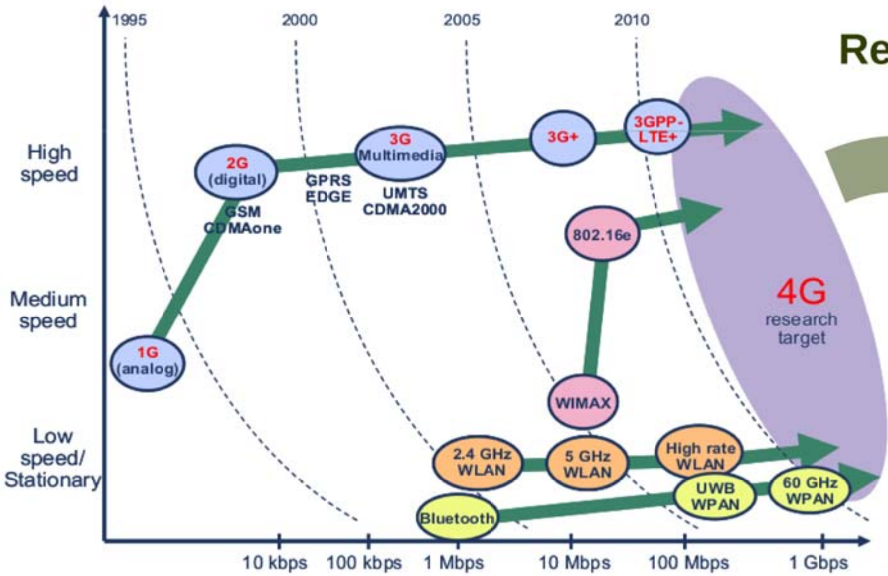
Platforms Evolution
Which New Architectures We Have to Target?

- From ad-hoc HW... to generic HW

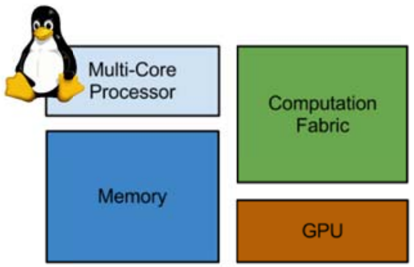


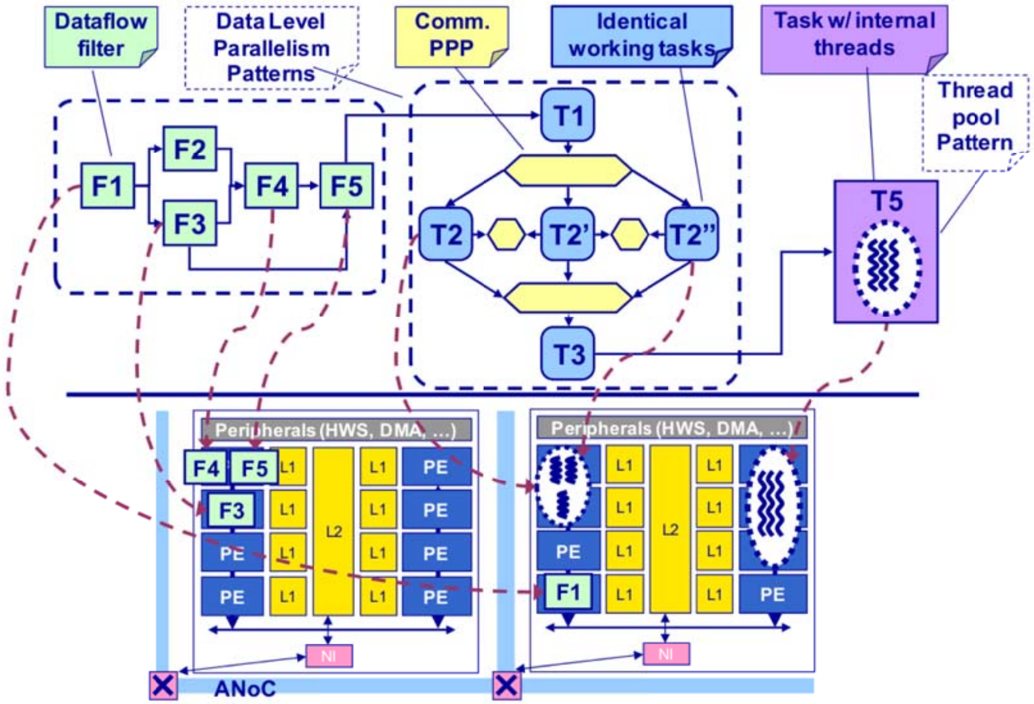
Platforms Evolution
How They can Be Exploited?

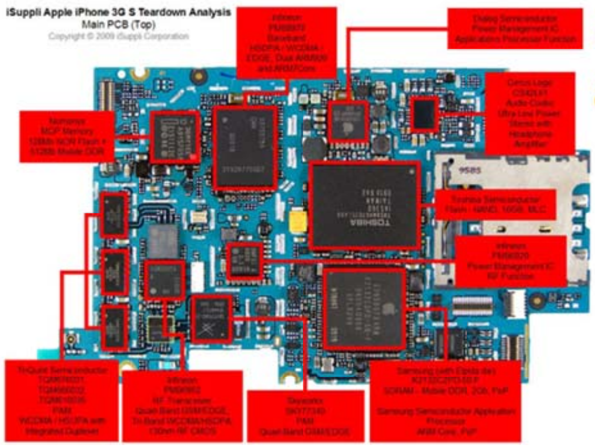
- Example: Software Defined Radios (SDR)



Re-Programmable approach







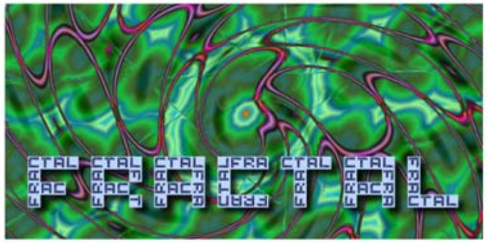
Classical approach

Platforms Evolution
Which Software Challenges?

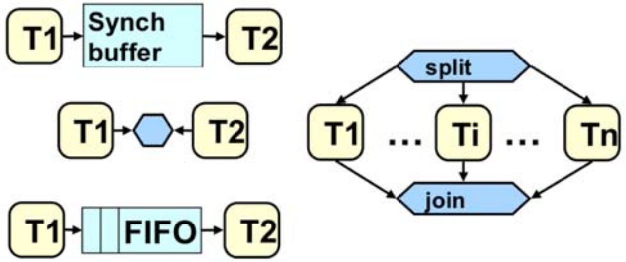
- Suppport for **parallel code development**
- Foster **reusable software components**
  - independent and parallelized SW modules (filters)
  - well defined interfaces to support composition (pipelines)
- New **programming paradigms**
  - to better support parallelized modules development
  - not binded to a specific target
    - “write one run anywhere”
- Usable development environments
  - high level of abstraction design of applications
  - target specific simulation and optimization support
  - support for multiple programming models

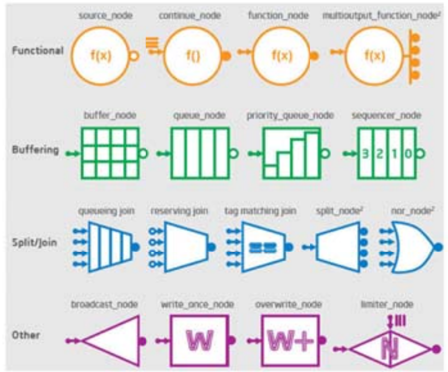
Platforms Evolution
Which Programming Paradigms?

- Proprietary and/or platform specific
  - Fractal
    - defined by OW2 Consortium
    - modular and extensible middleware
    - language agnostic (e.g. C, Java, .NET)
  - Native Programming Model
    - defined by STMicroelectronics
    - collection of primitives
      - to support decomposition
  - Thread Building Blocks (TBB)
    - defined by Intel
    - mostly targeting HPC
      - supporting just x86



http://fractal.ow2.org





http://threadingbuildingblocks.org

Platforms Evolution
Which Standards?

- OpenCL: “the” industrial standard

- OpenVX: the upcoming standard
  which introduces the concept of “task manager”

- Same principle used when playing with LEGOs

“collect, put together”

from Danish “leg godt” = “play well”

Platforms Evolution

Having a Data-Center in the Pocket

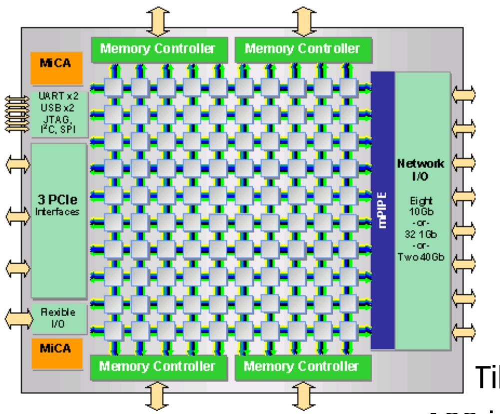
- Embedded is moving towards many-core architectures
  - Many similar computing elements
  - Complex applications are decomposed in parallel modules
- Device functionality is polymorphic
  - Depends on the programming
  - Can change at run-time adapting to the new scenario
- Resemble the HPC style
  - See last FP7 calls...

Barcelona Supercomputing Centre

10.240 processors

It's just a change of “scale factor”

Same benefits but “programmable”



Tilera Tile-Gx100
100 independent cores

Introduction to RTRM

overall view on goals, requirements and design

- Computing platforms convergence
    - *targeting both **HPC** and **high-end embedded** and mobile systems*
    - parallelism level ranging from few to hundreds of PEs
        - thanks to silicon technology progresses
- Emerging new set of non-functional constraints
    - **thermal** management, system **reliability** and fault-**tolerance**
        - area and power are typical design issues
    - *embedded systems are loosing exclusiveness*

effective resource management policies required to properly exploit modern computing platforms

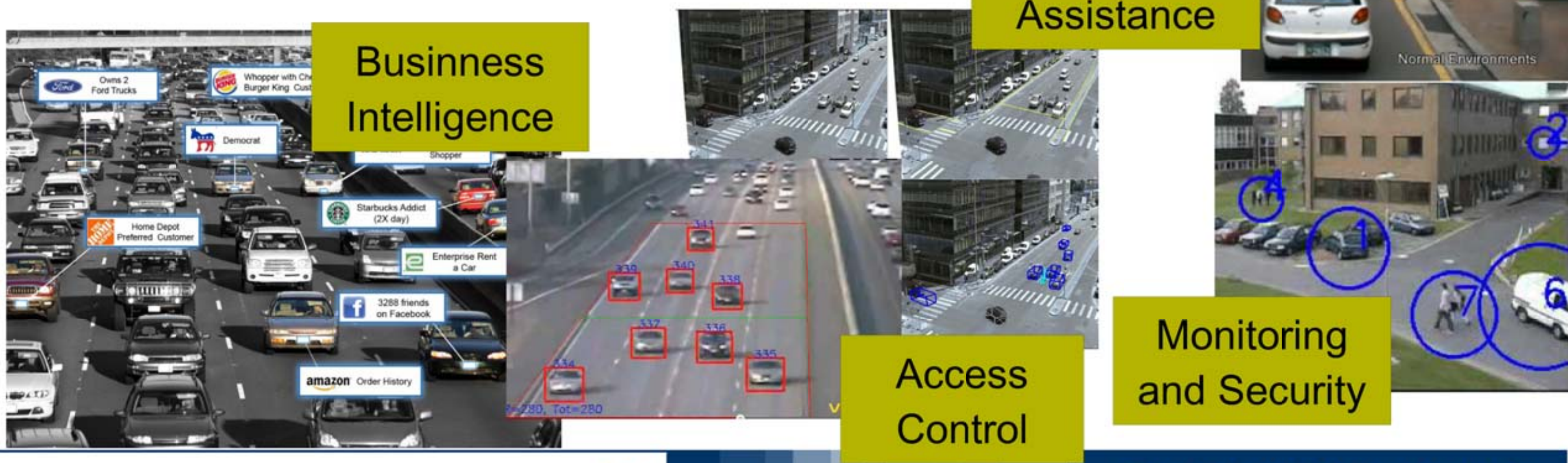
- Run-Time Resources Management (RTRM) is about

  *finding the optimal **tradeoff** between*

  QoS requirements and resources availability

- Target scenario
  - Shared HW resources
    - *upcoming many-core devices are complex systems*
      - *process **variations** and run-time issues*
  - Mixed SW workloads
    - *resources sharing and competition*
      - among applications with different and **time-varying** requirements
- Simple solutions are required
  - support for frequently *changing* use-cases
  - suitable for both *critical and best-effort* applications

- Many-core platforms enable a new set of applications
  - computer vision is just one of the main interesting
- Multi-functional embedded devices are widespread
  - concurrently running applications
  - different criticality
  - time-varying requirements



- 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

Introduction to RTRM
How we compare?

- Different approaches targeting resources allocation
  - Linux scheduler extensions
    - mostly based on adding new scheduler classes [2,4,7]

force the adoption of a customized kernel

  - Virtualization
    - *Hypervisor acting as a global system manager*
    - *Both commercial and open source solutions*
      - *Commercial: e.g. OpenVZ, VServer, Montavista Linux; Open: e.g. KVM, Linux Containers*

require HW support on the target system

  - User-space approaches
    - *more portable solutions [3,6,11]*

mostly limited to CPU assignment

[2] Bini et. al., **"Resource management on multicore systems: The actors approach"**. Micro 2011.
[3] Blagodurov and Fedorova, **"User-level scheduling on numa multicore systems under linux"**, Linux Symposium 2011.
[4] Fu and Wang., **"Utilization-controlled task consolidation for power optimization in multi-core real-time systems"**. RTCSA 2011.
[6] Hofmeyr et. al.,. **"Load balancing on speed"**. PpoPP 2010.
[7] Li et. al., **"Efficient operating system scheduling for performance-asymmetric multi-core architectures"**. SC 2007.
[11] Sondag and Rajan, **"Phase-based tuning for better utilization of performance-asymmetric multicore processors"**. CGO 2011.

- Different approaches targeting resources allocation
  - Linux scheduler extensions
    - r
    - *l kernel*
  - Virtu
    - *H*
    - *Both commercial and open source solutions*
      - *Commercial: e.g. OpenVZ, VServer, Montavista Linux; Open: e.g. KVM, Linux Containers*
    - ***require HW support on the target system***
  - User-space approaches
    - *more portable solutions [3,6,11]*
    - ***mostly limited to CPU assignment***

More dynamic usage of **Linux Control Groups** to manage **multiple resources** with a **portable** and **modular** RTRM running in user-space

[2] Bini et. al., **"Resource management on multicore systems: The actors approach"**. Micro 2011.
[3] Blagodurov and Fedorova, **"User-level scheduling on numa multicore systems under linux"**, Linux Symposium 2011.
[4] Fu and Wang., **"Utilization-controlled task consolidation for power optimization in multi-core real-time systems"**. RTCSA 2011.
[6] Hofmeyr et. al.,. **"Load balancing on speed"**. PpoPP 2010.
[7] Li et. al., **"Efficient operating system scheduling for performance-asymmetric multi-core architectures"**. SC 2007.
[11] Sondag and Rajan, **"Phase-based tuning for better utilization of performance-asymmetric multicore processors"**. CGO 2011.

The Barbeque Approach to RTRM
an overall view on proposed tool architecture

The BarbequeRTRM
Overall Contributions

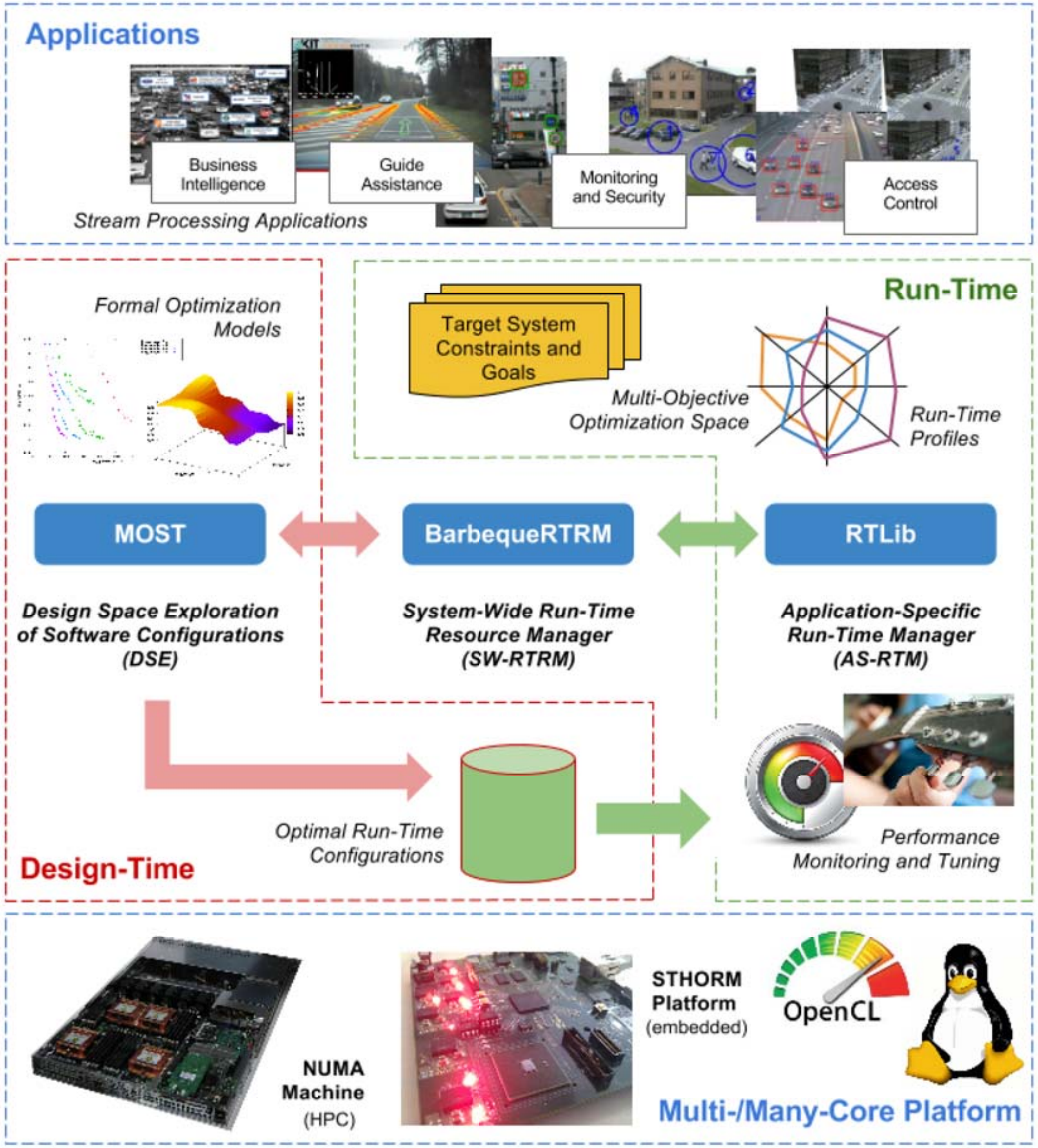
- **Methodology** to support system-wide run-time resource management
  - exploiting design-time information
  - hierarchical and distributed control

http://www.2parma.eu

- BarbequeRTRM **Framework**
  - multi-objective optimization strategy
  - easily portable and modular design
  - run-time tunable and scalable policies
  - open source project

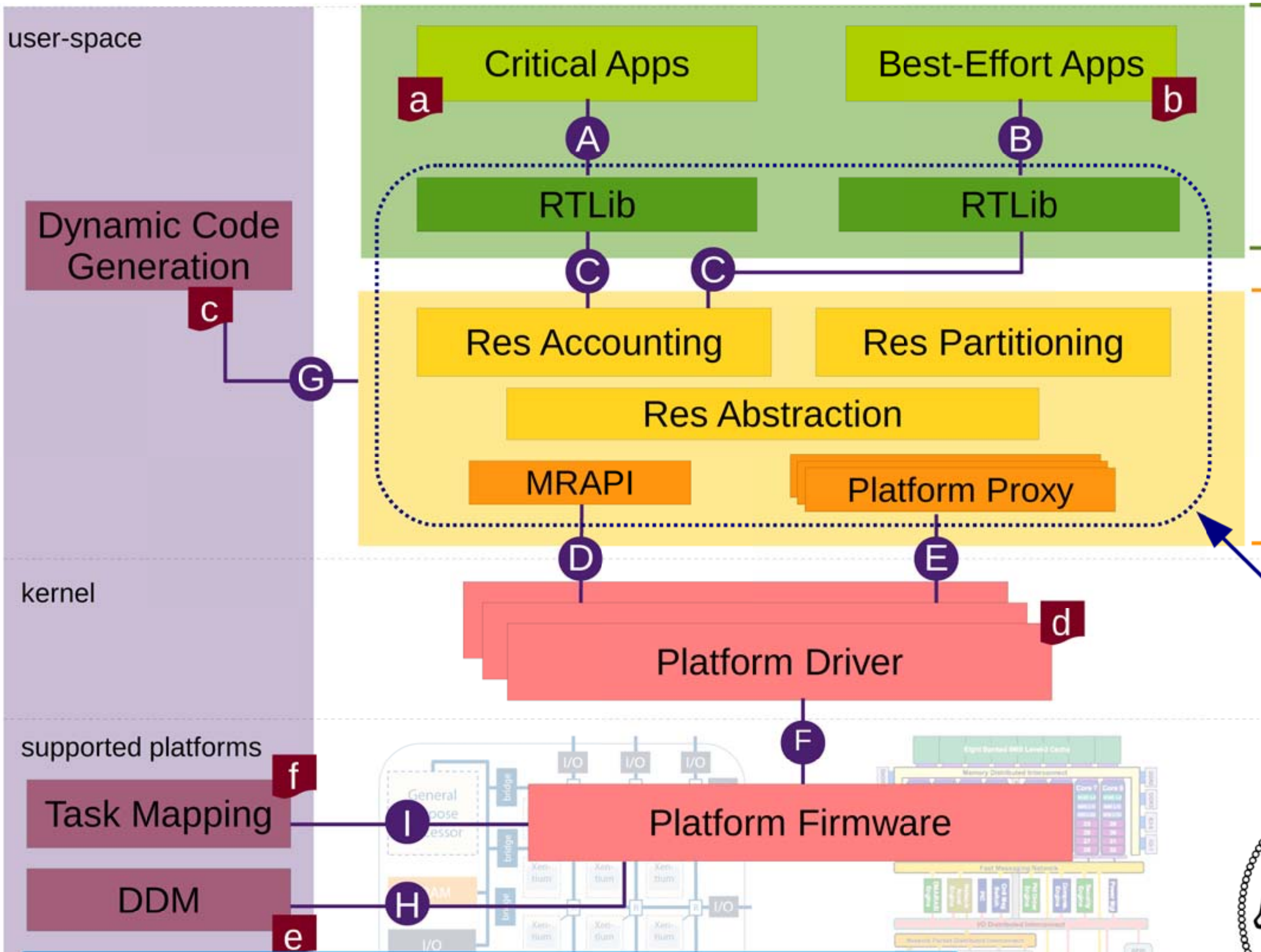
http://bosp.dei.polimi.it

The BarbequeRTRM
A Bird Eye View on the Proposed Approach



- Track run-time variabilities
  - application requirements
  - resources availabilities
- Overhead contingency
  - design-time profiling
  - run-time optimization
- Support different granularity
  - system-wide optimization
  - application-specific tuning
- Integrated work-flow
  - single framework to support both design-time and run-time

The BarbequeRTRM
Overall View on Run-Time Resource Management



Application-Specific RTM

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

System-Wide RTRM

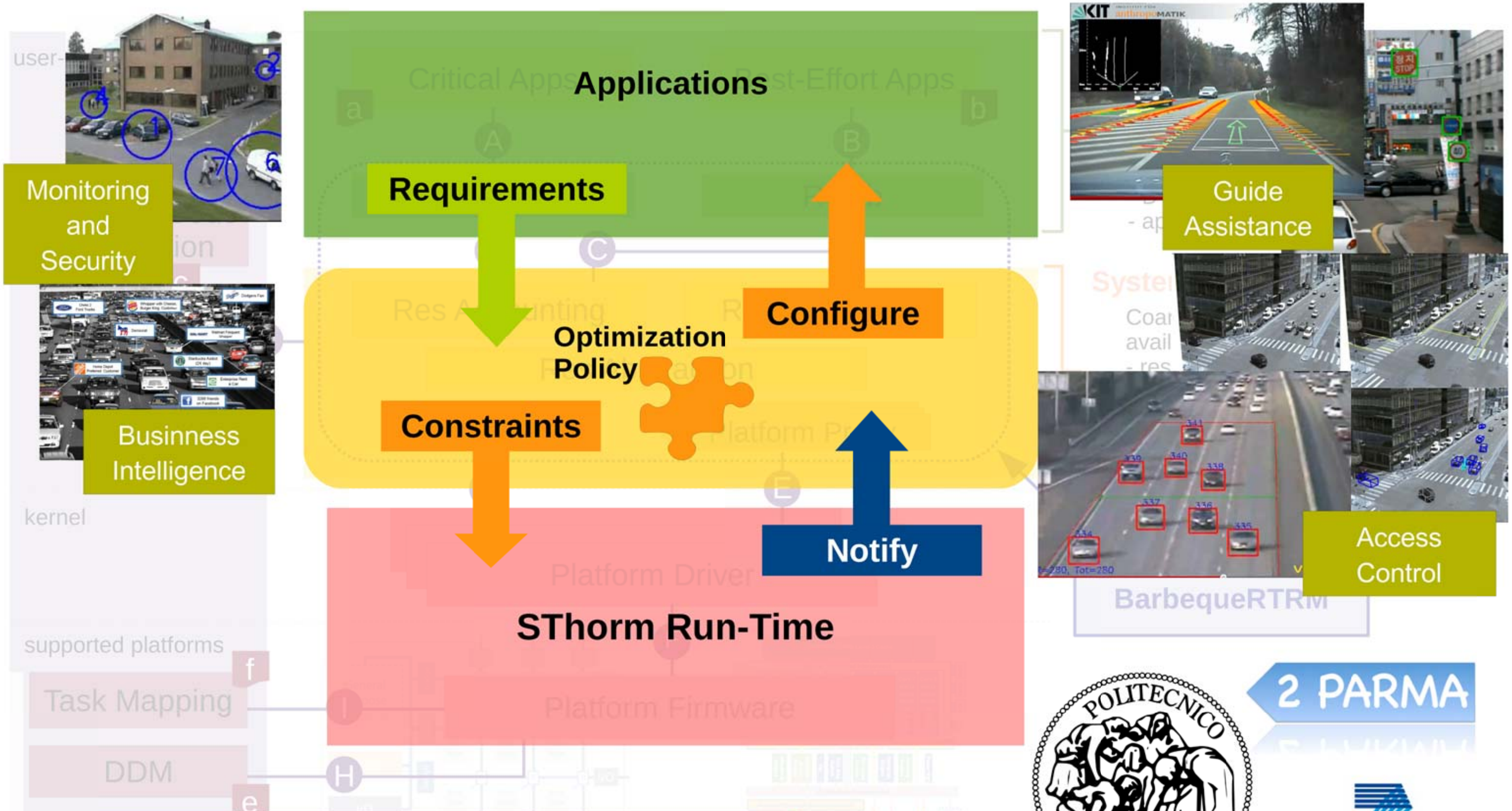
Coarse grained control on platform available resources:
- resource accounting, partitioning and abstraction
- high-level HW events handling e.g., critical conditions, faults...
- manage applications priorities
- power/thermal “coarse tuning”

BarbequeRTRM

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

The BarbequeRTRM

Overall View on Run-Time Resource Management



[1] Bellasi et.al., **"A RTRM proposal for multi/many-core platforms and reconfigurable applications"**. ReCoSoC 2012.

The BarbequeRTRM
Example: Multi-Core NUMA Platforms

Congested workloads

Regular Workload

Critical Apps — a — A
Best-Effort Apps — b — B

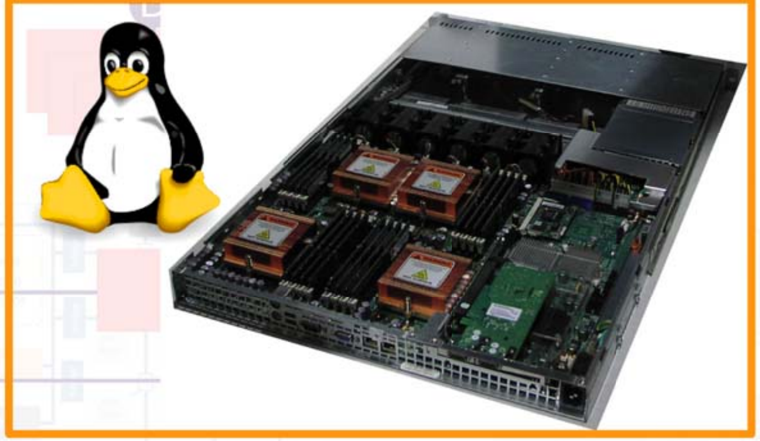
RTLib | RTLib

C | C

Res Accounting | Res Partitioning

Res Abstraction

CGroups



CGroups based resources abstraction layer

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

Extend advanced and efficient resources control capability offered by modern Linux Kernels

with suitable resources partitioning policies

running in user-space

supported platforms
Task Mapping
DDM

The BarbequeRTRM
Example: Many-Core STHorm Platform

Congested workloads

Regular Workload

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

Extend SThorm resident run-time scheduler capability offered by current p12runtime

with suitable resources partitioning policies

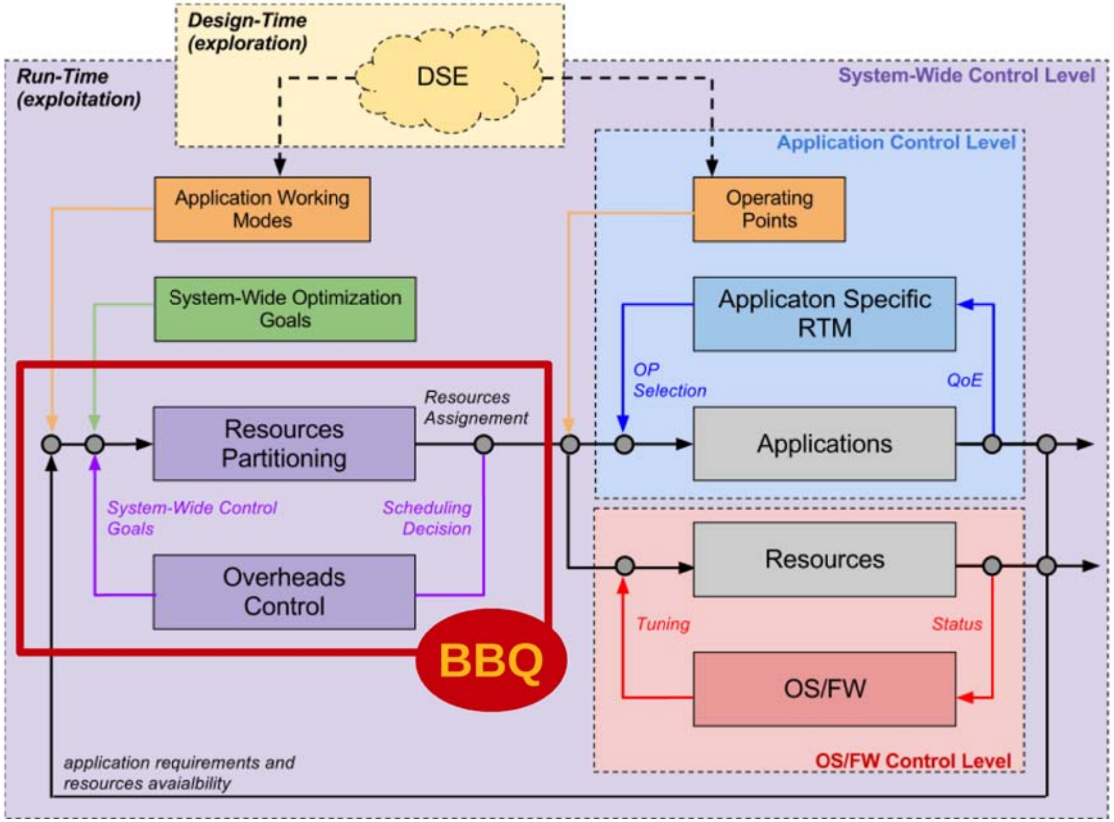
Managed by a user-space daemon

supported platforms
Task Mapping
DDM

Memory mapped resources abstraction layer

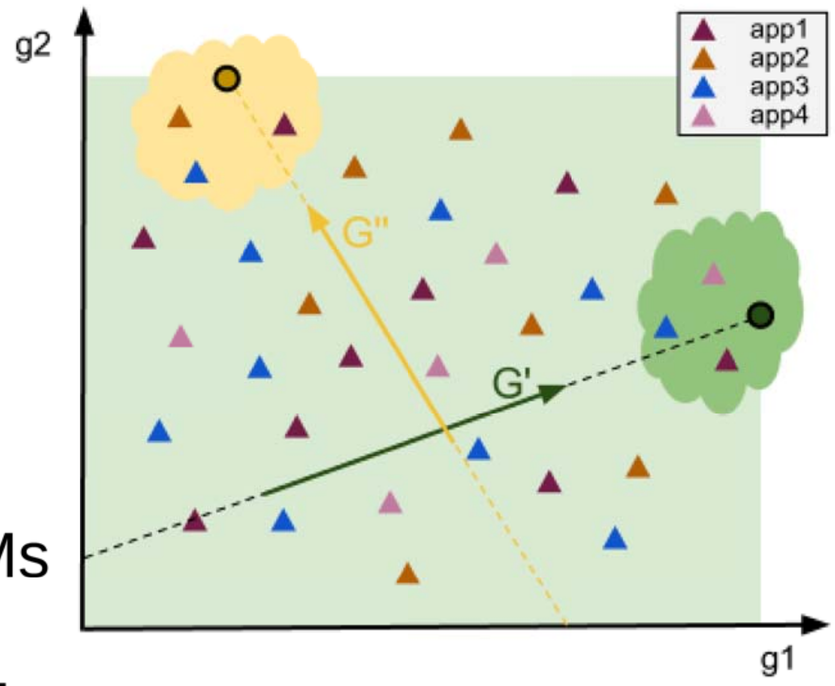
The Proposed Control Solution
Distributed Hierarchical Control

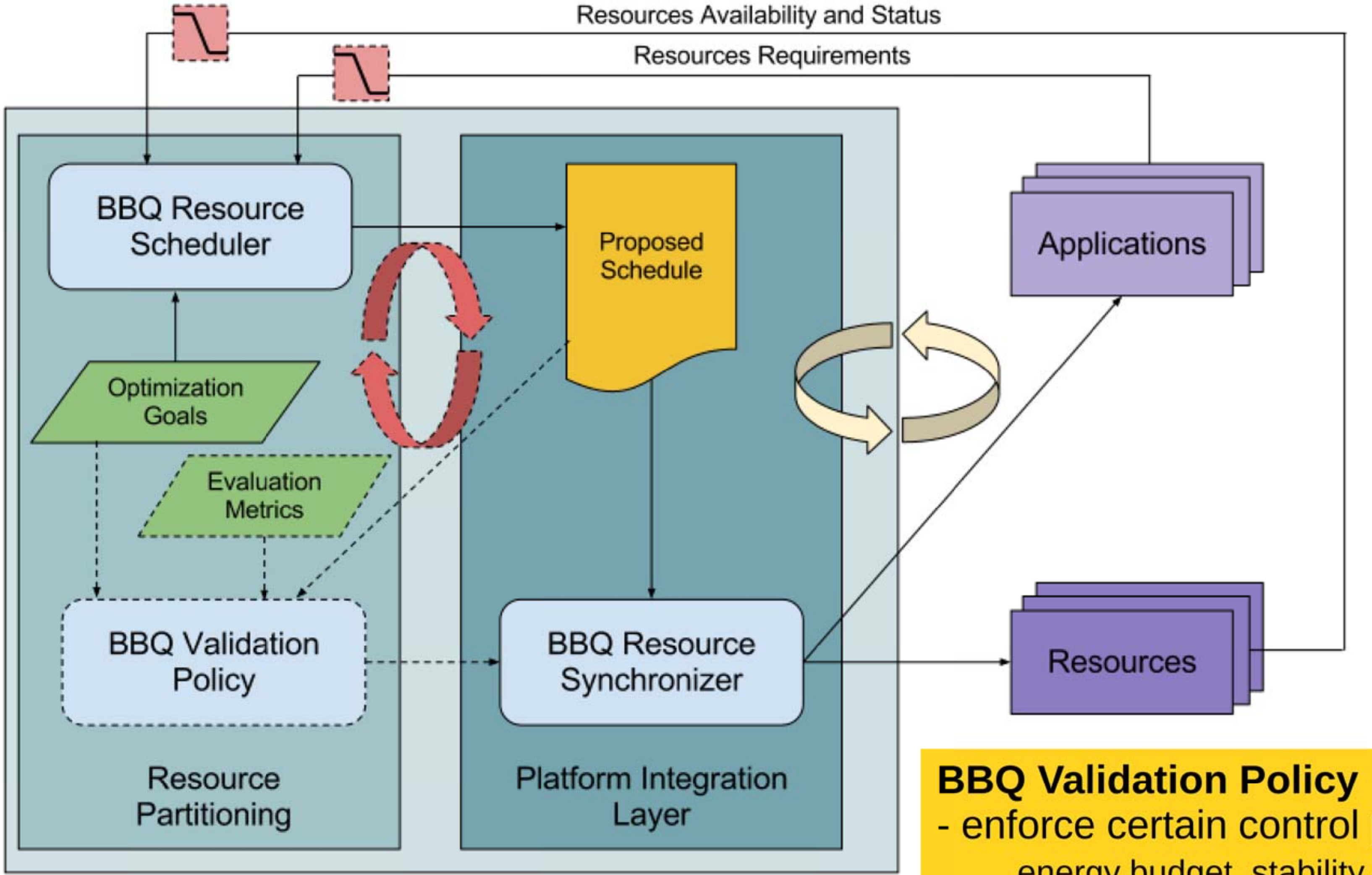
- Different subsystems have their own control loop (CL)
  - System-wide level (resources partitioning, system-wide optimization, ...)
  - Application specific (application tuning, dynamic memory management, ...)
  - Firmware/OS level (F/V control, thermal alarms, resource availability, ...)
- FF closed CL
  - using OP and AWM
- Optimal
  - user defined goal functions including overheads
- Robust
- Adaptive



Scheduling Policy
YaMS - A modular multi-objective scheduler

- Introduction of a new modular policy (YaMS)
  - partition available resources (R) on applications (A)
    - *considering A priorities and R "residual" availabilities*
  - multi-objective optimization
    - *support a set of tunable goals*
      - *DONE: performances, overheads, congestion, fairness*
      - *WIP: stability, robustness, thermal and power*
    - *increase overall system value*
      - *considering discrete and tunable improvements*
- LP theory, MMKP heuristic
  - **promote** scheduling of some AWMs
    - *which improve optimization goals*
  - **demote** scheduling of others AWMs
    - *which degrade solution metrics*
      - *e.g. stability and robustness*

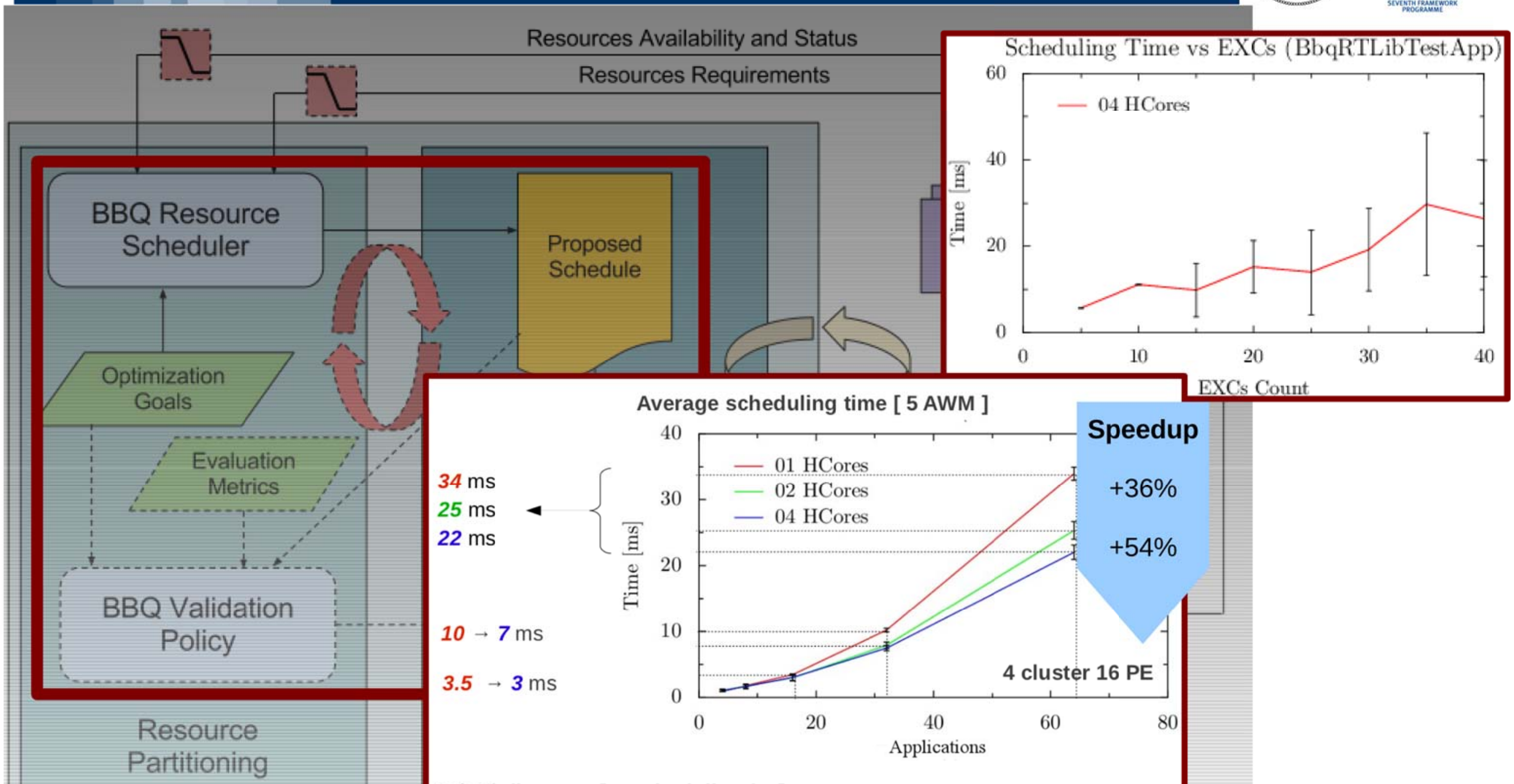




BBQ Validation Policy

- enforce certain control properties
  - energy budget, stability and robustness
- authorize resources synchronization

Scheduling Policy
System-Wide Controller – Scalable and "Fast Response"

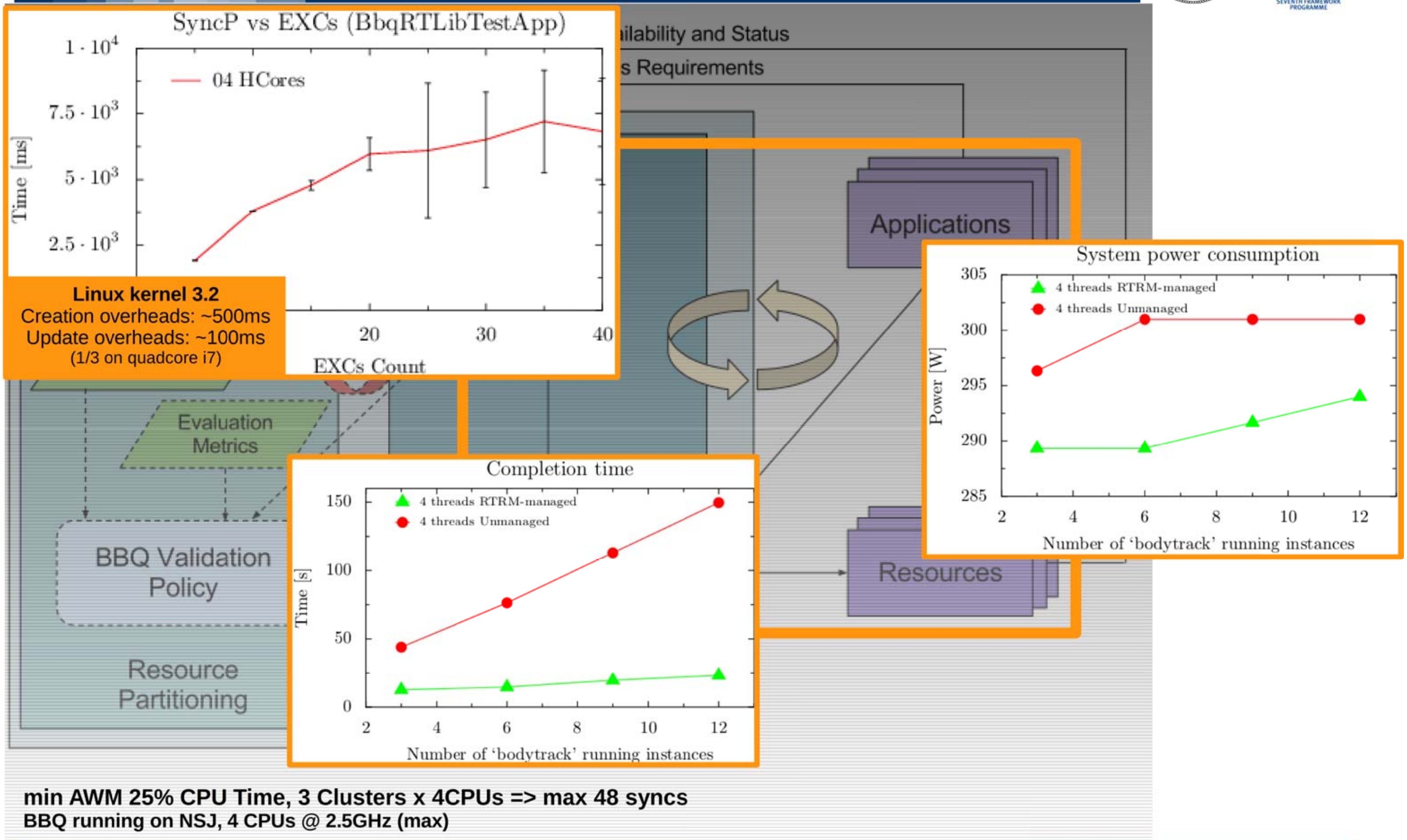


Scheduling rate [n. scheduling / s]

16 applications → ~ ***300***
32 applications → ***100 .. 130***
64 applications → ***30 .. 45***

Scheduling Policy

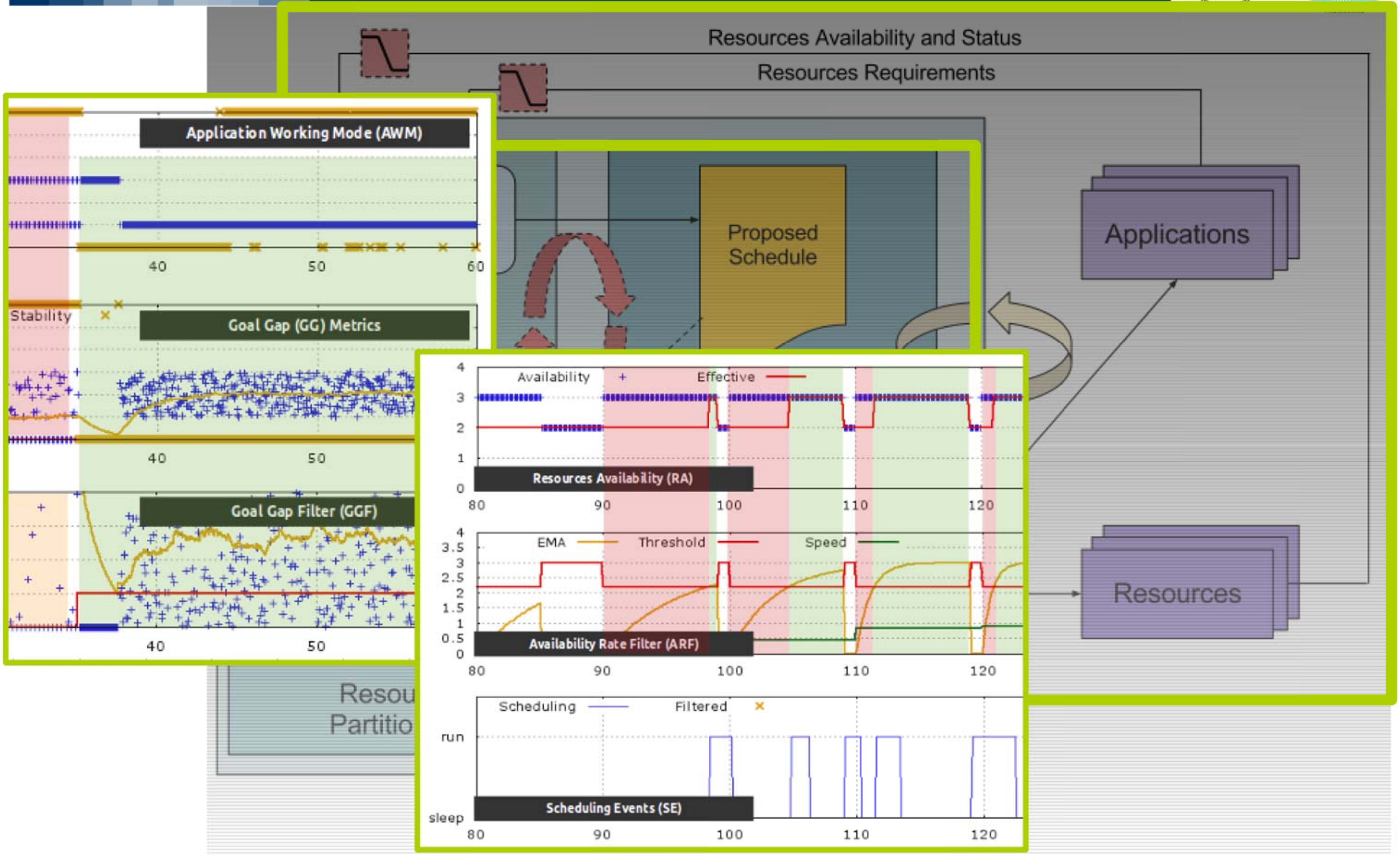
System-Wide Controller – Scalable and "Fast Response"



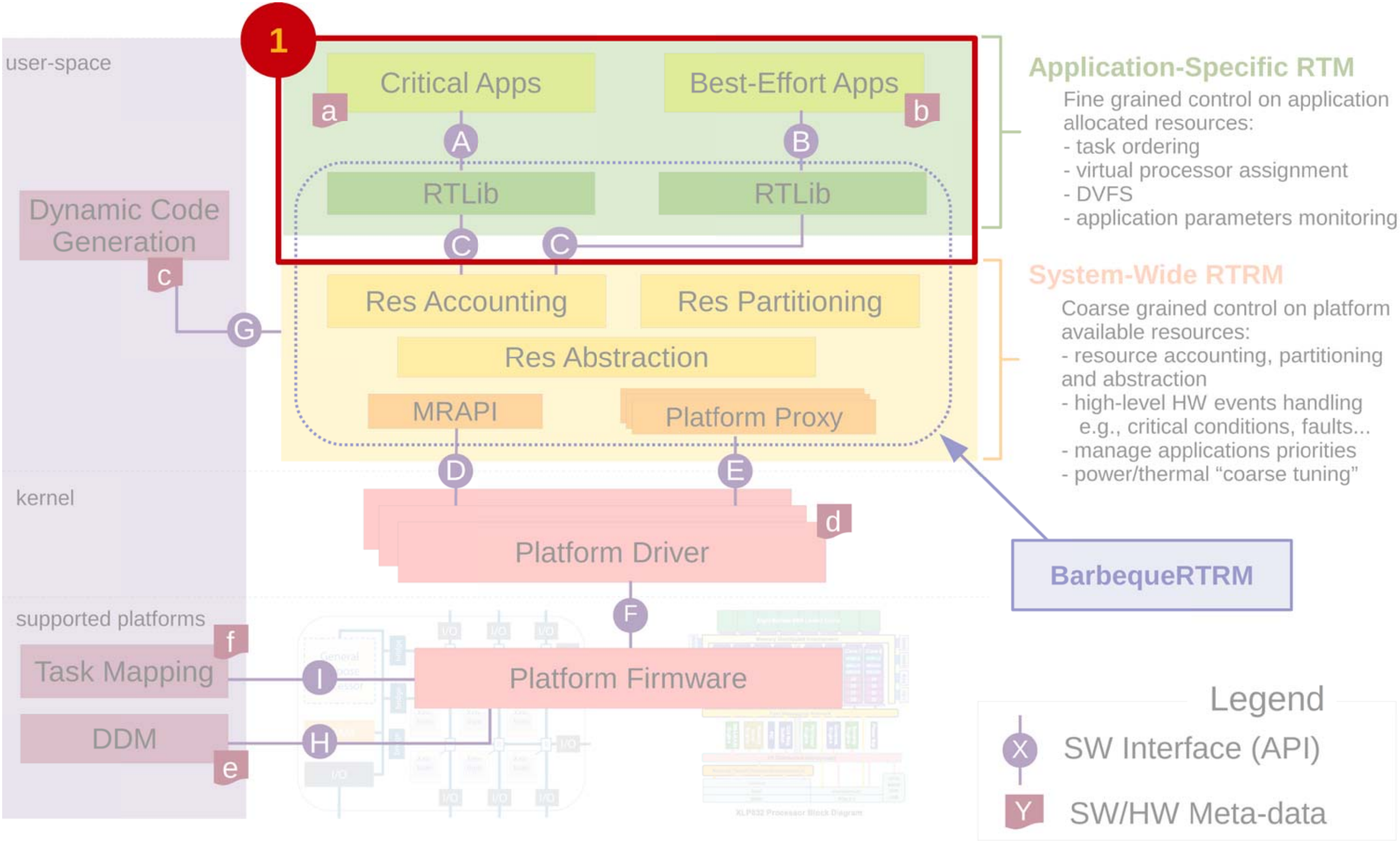
min AWM 25% CPU Time, 3 Clusters x 4CPUs => max 48 syncs

BBQ running on NSJ, 4 CPUs @ 2.5GHz (max)

Scheduling Policy
System-Wide Controller – Grant Stability and Robustness



Application Integration Layer
System-Wide RTRM: RTLib details



Application-Specific RTM

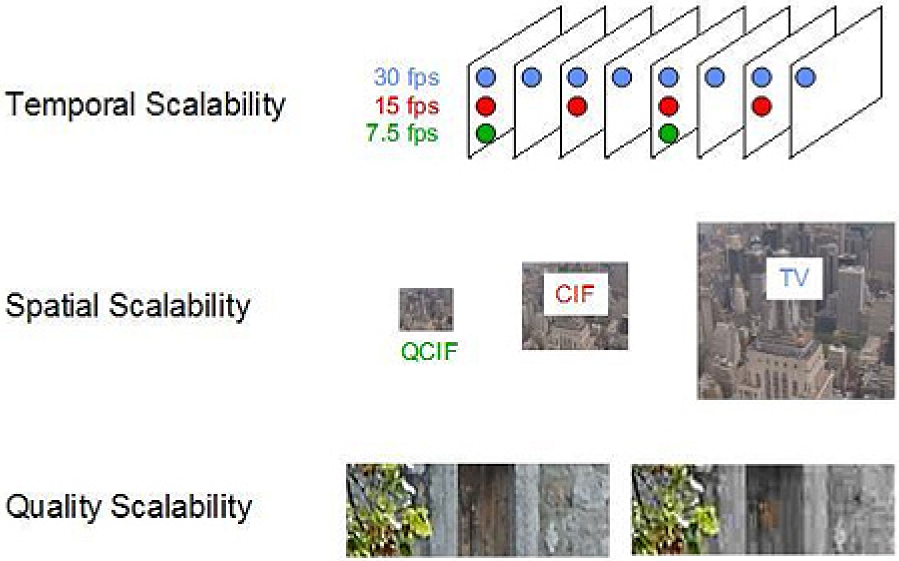
Fine grained control on application allocated resources:
- task ordering
- virtual processor assignment
- DVFS
- application parameters monitoring

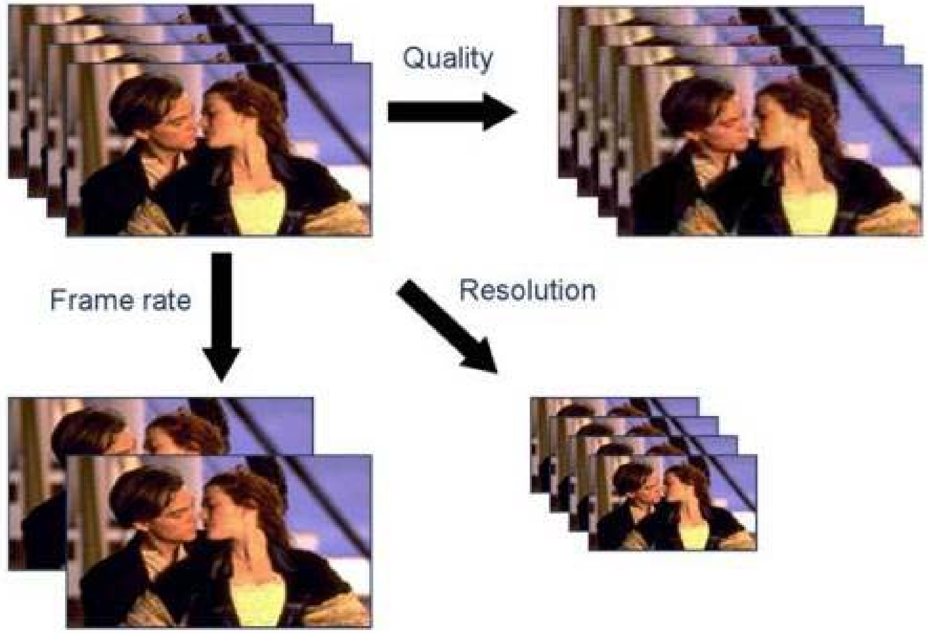
System-Wide RTRM

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"

Application Integration Layer
Target applications (1/2)

- Run-time **reconfigurable** workloads
  - e.g. Scalable Video Coding (SVC)
    - *single input stream, different decoding configurations*

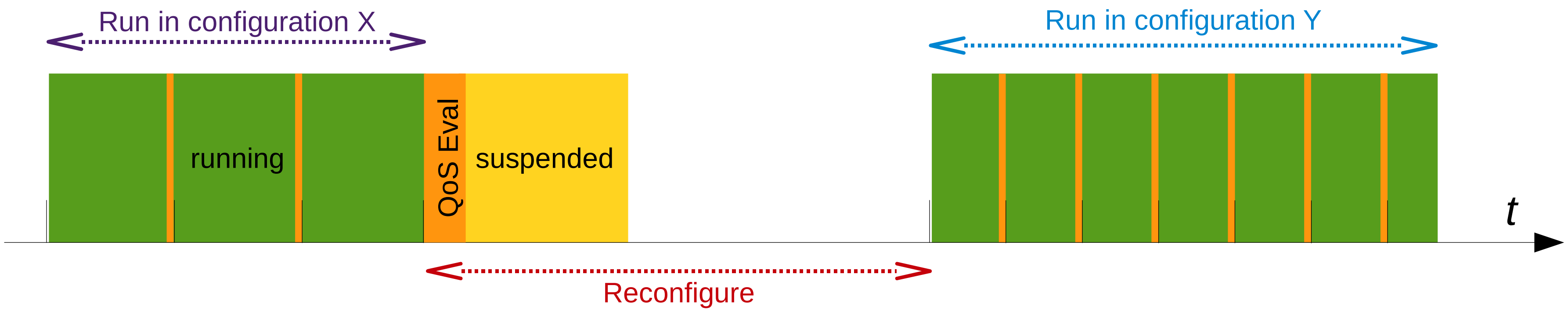




*Different **decoding profiles** which corresponds*
*to different **quality-vs-performances** requirements*

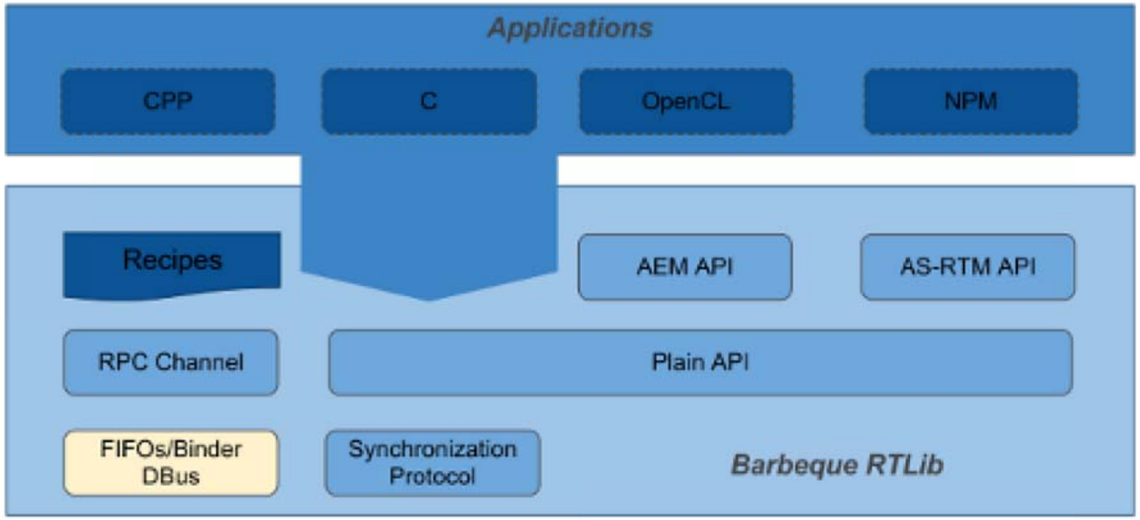
2PARMA Project Demo - BarbequeRTRM v0.6 (Angus)
http://youtu.be/B1TDNbtIKC8

- Stream processing applications
  - which means not only multimedia processing
    - *e.g. packet sniffing and analysis, pattern matching, ...*
- Well defined **Abstract Execution Model** (AEM)
  - loop of actions, until no more workload to process
    - *Setup, Configure, Running, Monitor*



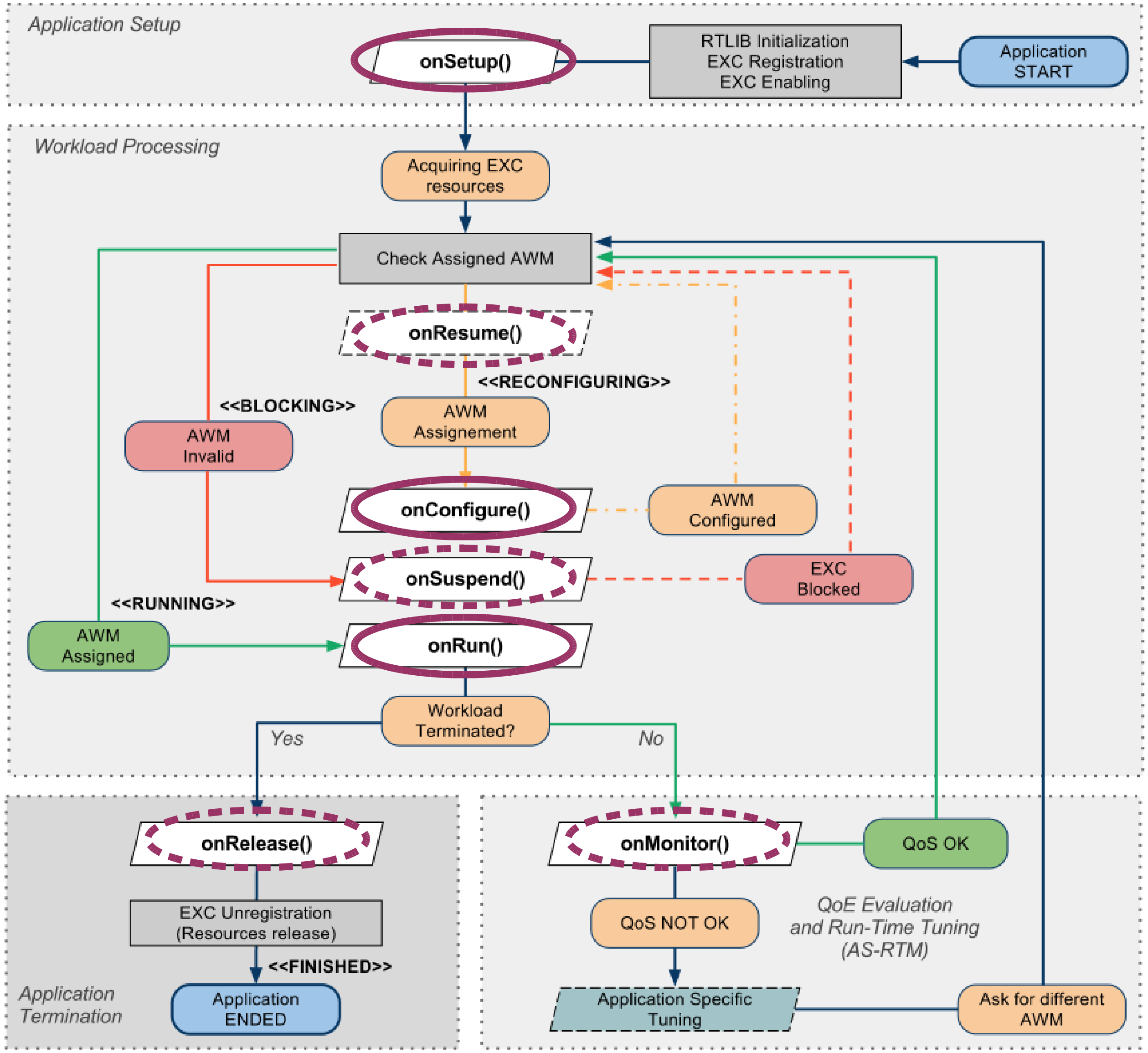
Application Integration Layer
Run-Time Library (RTLib)

- Defines the (expected) application behavior
  - loop of actions, until no more workload to process
- Abstract the communication channel
  - using “threaded FIFOs”, (WIP) Binder support on Android
- Provides APIs at **three different abstraction levels**
  - Plain API, AEM API and AS-RTM API
- Hides the **Synchronization-Protocol** details



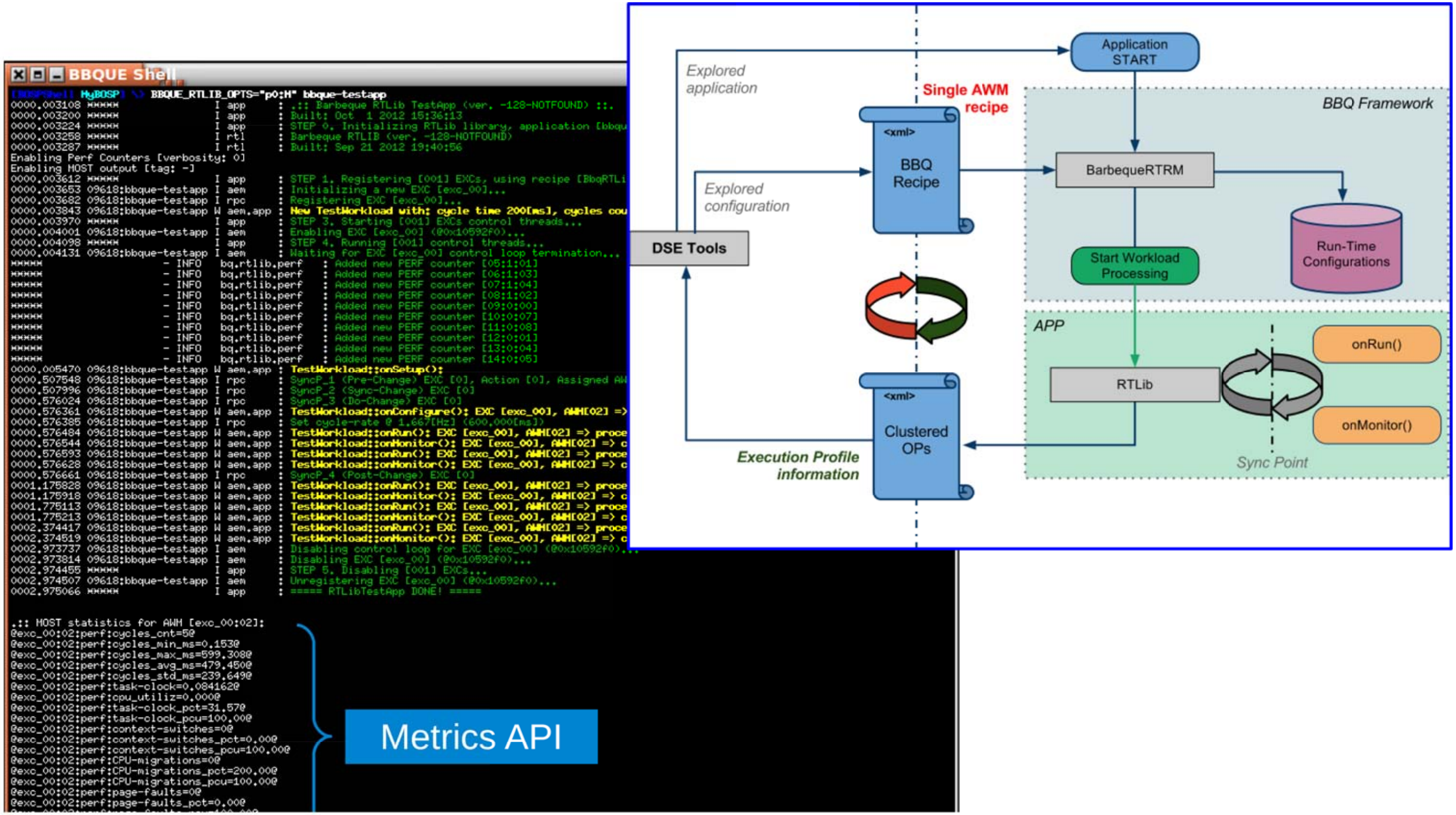
AEM Abstract API

- **callbacks** based
  with default implementations
- hide all the RTM boilerplate code



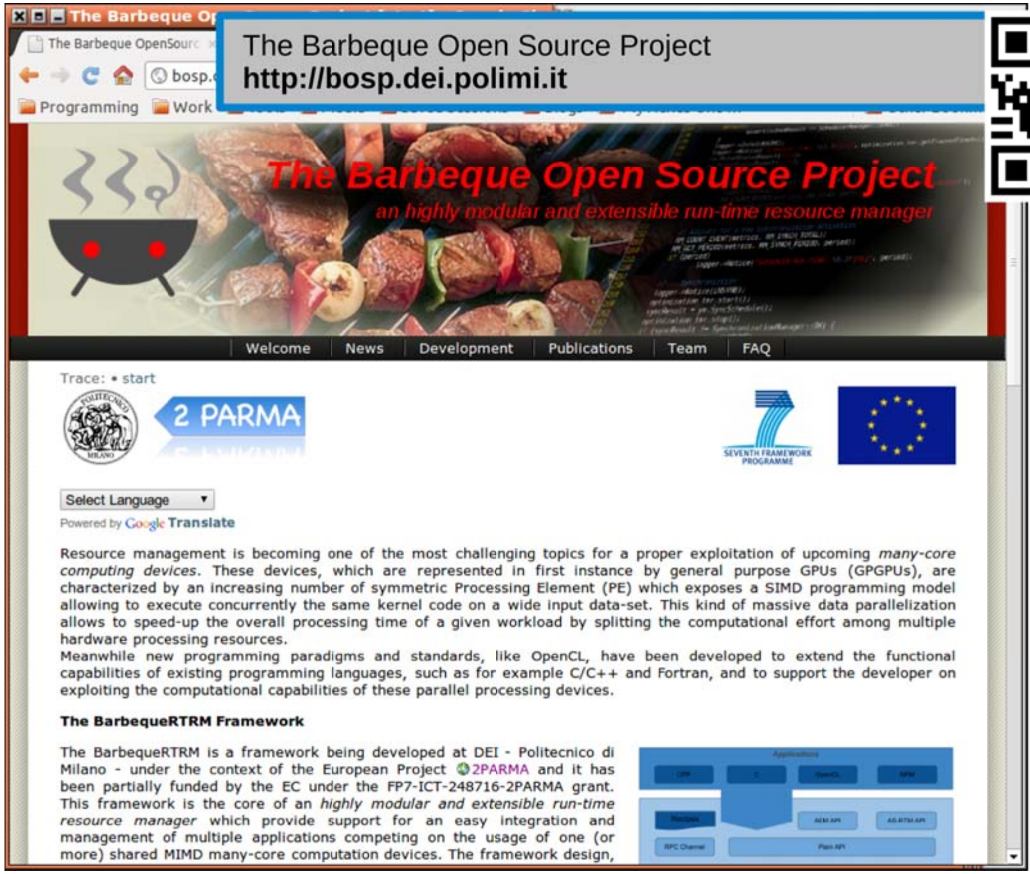
Applications Integration

MOST DSE Tool integration



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, RTLib*

Framework Tools
- *PyGrill (loggrapher), ...*

Contributions
- *Tutorials, demo*

Public GIT repository
https://bitbucket.org/bosp

- 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

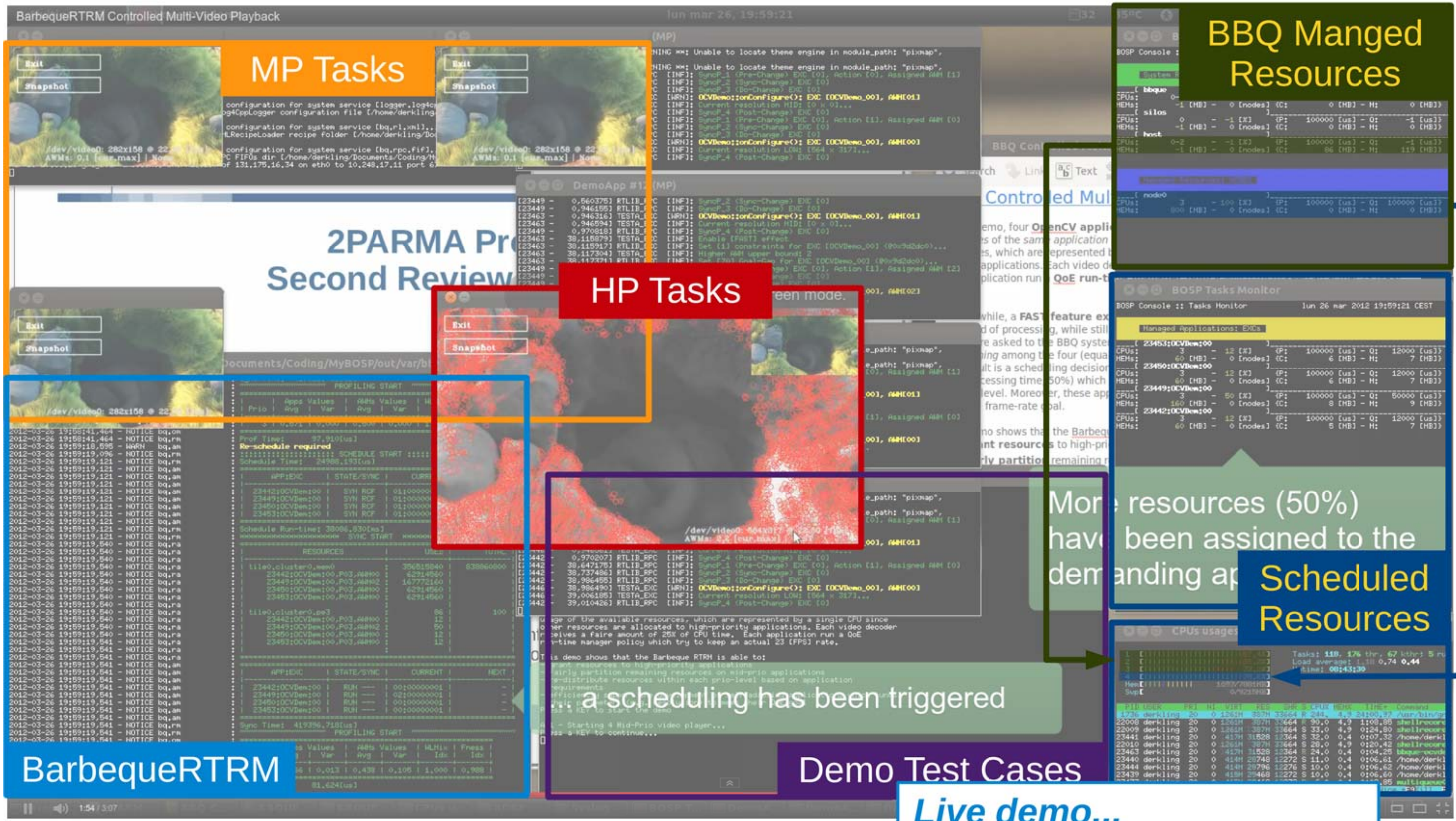
Bellasi et.al., "**A RTRM proposal for multi/many-core platforms and reconfigurable applications**". ReCoSoC 2012.

Official Project Website

http://bosp.dei.polimi.it

The BarbequeRTRM *v0.8 (Betty Bacon)*

Outline of the Demo Video



Live demo...
(or visit http://youtu.be/Hcz1ob23WWA)

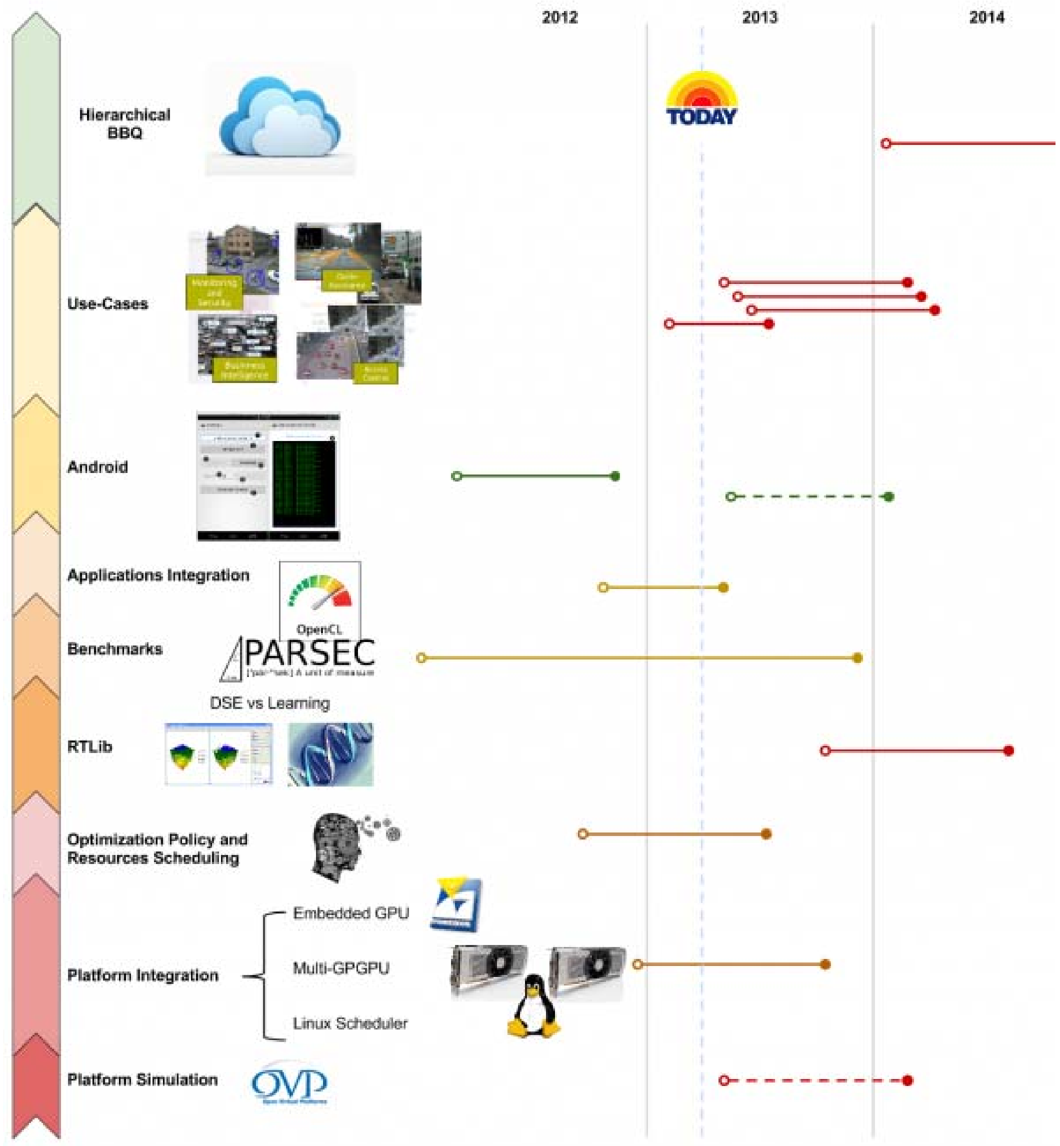
The BarbequeRTRM Framework
Conclusions and Future Works

- Framework for System-Wide RTRM
  - **flexibility and scalability** of the RTRM strategy
    - *thanks to its hierarchical and distributed control structure*
  - **acceptable overheads** for real usage scenarios
    - *including those with variable workload*
  - tunable **multi-objective optimization** policies
    - *to cope with several design constraints and goals*
      - *e.g., performance, power, thermal and reliability, ...*
  - promising results in terms of **performance improving** and **power consumption reduction**
    - *for a highly parallel workload, on a NUMA multi-core architecture*
  - availability of a simple API interface
    - *making straightforward for the programmers to take full advantages from framework services*

The BarbequeRTRM Framework
Future Works

Wide spectrum of activities

covering different abstraction level



Thanks for your attention!

Under negotiation in FP7

Strep – run time for reliability and QoS guaranteed. HPC and ES synergy

IP – mixed criticalities, WSN+cloud

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

Prof. William FORNACIARI

home.deib.polimi.it/fornacia

william.fornaciari@polimi.it

The POLIMI team

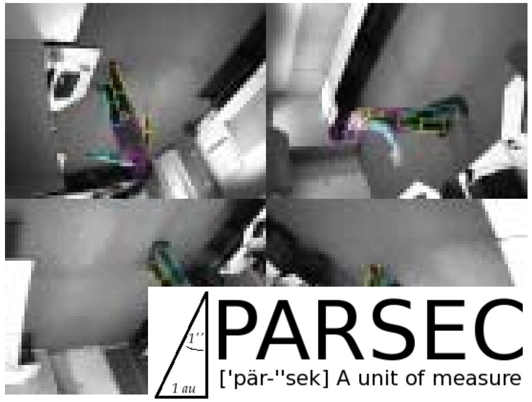
BBQ Fmk: P.Bellasi, G.Massari
Metrics for thermal: D.Zoni, S.Corbetta, F.Terraneo
New runtime directions:, L.Rucco, C.Caffarri, C.Brandolese
DSE: C.Silvano, G.Palermo, V.Zaccaria, E.Paone
Progr. Paradigms: G.Agosta, Scandale

http://bosp.dei.polimi.it

Backup Slides

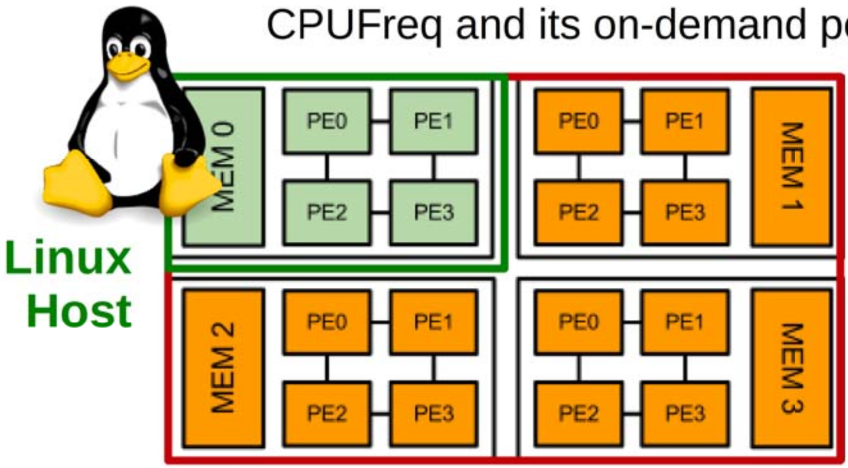
Experimental Setup
Hardware Platform and Workloads

- Workloads: increasing number of concurrently running applications
  - Bodytrack (BT) (PARSEC v2.1)
    - *modified to be run-time tunable and integrated with the BarbequeRTRM*



https://bitbucket.org/bosp/benchmarks-parsec

- Platform: Quad-Core AMD Opteron 8378
  - 4 core host partition, 3x4 CPUs accelerator partition
    - running up to 2.8GHz , 16 Processing Elements (PE)
    - CPUFreq and its on-demand policy



Goal: assess framework capability to efficiently manage resources on increasingly congested workload scenarios

Experimental Setup
Metrics Collection

- Compare Bodytrack original vs integrated version
  - using same maximum amount of thread
    - the BBQ Managed version could reduce this number at Run-Time
- Original version controlled by Linux scheduler, integrated version managed by BarbequeRTRM

- Performances profiling
  - using standard frameworks

IPMI Interface for system-wide power consumption [W]

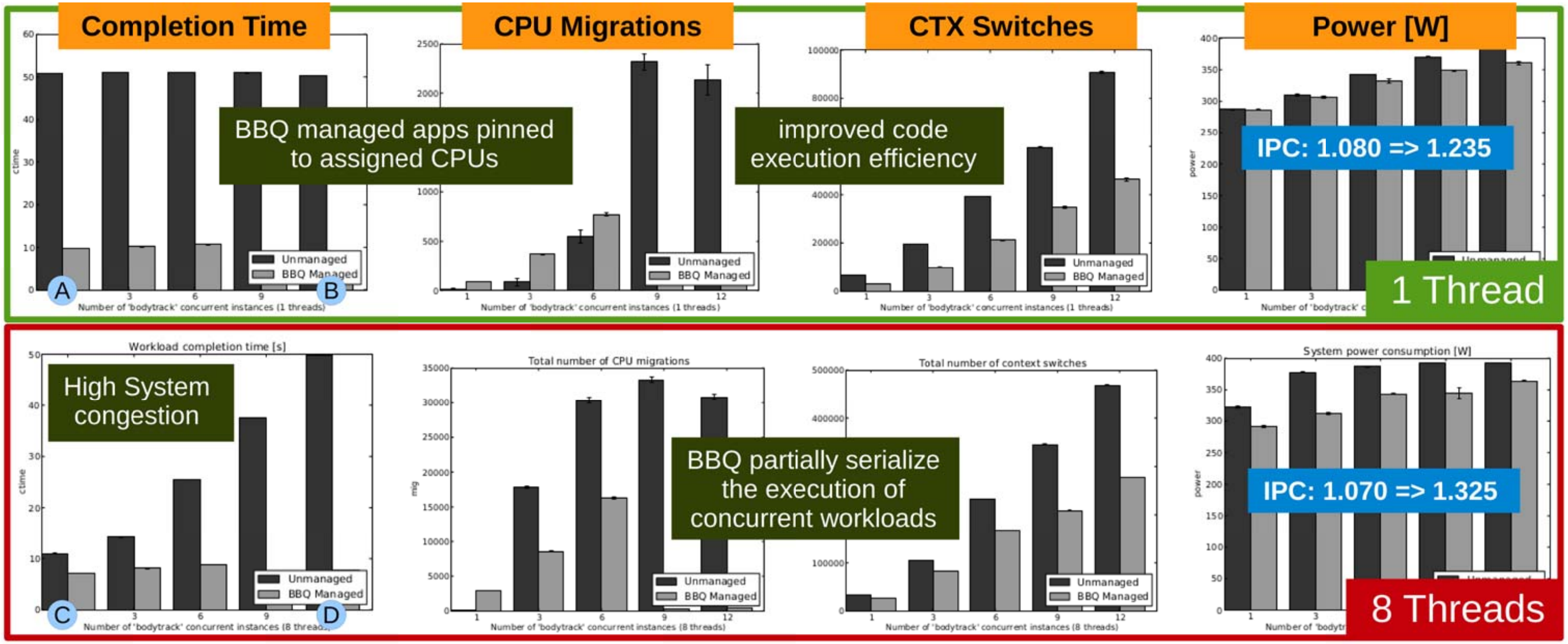
Using Linux *perf* framework to collect HW/SW performance counter

Goal*	Description
CTIME	Time [s] - Workload completion time [s]
POWER	Power [W] - System power consumption [W]
TASK-CLOCK	Ticks - Task clock ticks
CTX	Context-Switches - Total number of context switches
MIG	Migrations - Total number of CPU migrations
PF	Page-Faults - Total number of page faults
CYCLES	Cycles - Total number of CPU cycles
FES	Front-End Stalls - Total number of front-end stalled-cycles
FEI	Front-End Idles - Total number of front-end idle-cycles
BES	Back-End Stalls - Total number of back-end stalled-cycles
BEI	Back-End Idles - Total number of back-end idle-cycles
INS	Instructions - Total number of executed instructions
SCPI	SPC - Effective Stalled-Cycles-per-Instruction
B	Branches - Total number of branches
B-RATE	Branches-Rate - Effective rate of branch instructions
B-MISS	Branch-miss - Total number of missed branches
B-MISS-RATE	Branch-miss Quota - Effective percentage of missed branche
GHZ	GHz - Effective processor speed
CPU-USED	CPUs utilized - CPUs utilization
IPC	IPC - Effective Instructions-per-Cycles

(*) The lower the better, for all metrics but the IPC

Results

Workload Burst Performance Comparison



Reduced OS overhead
Improved code efficiency

> x1.3 faster

Up to x6 more energy efficient

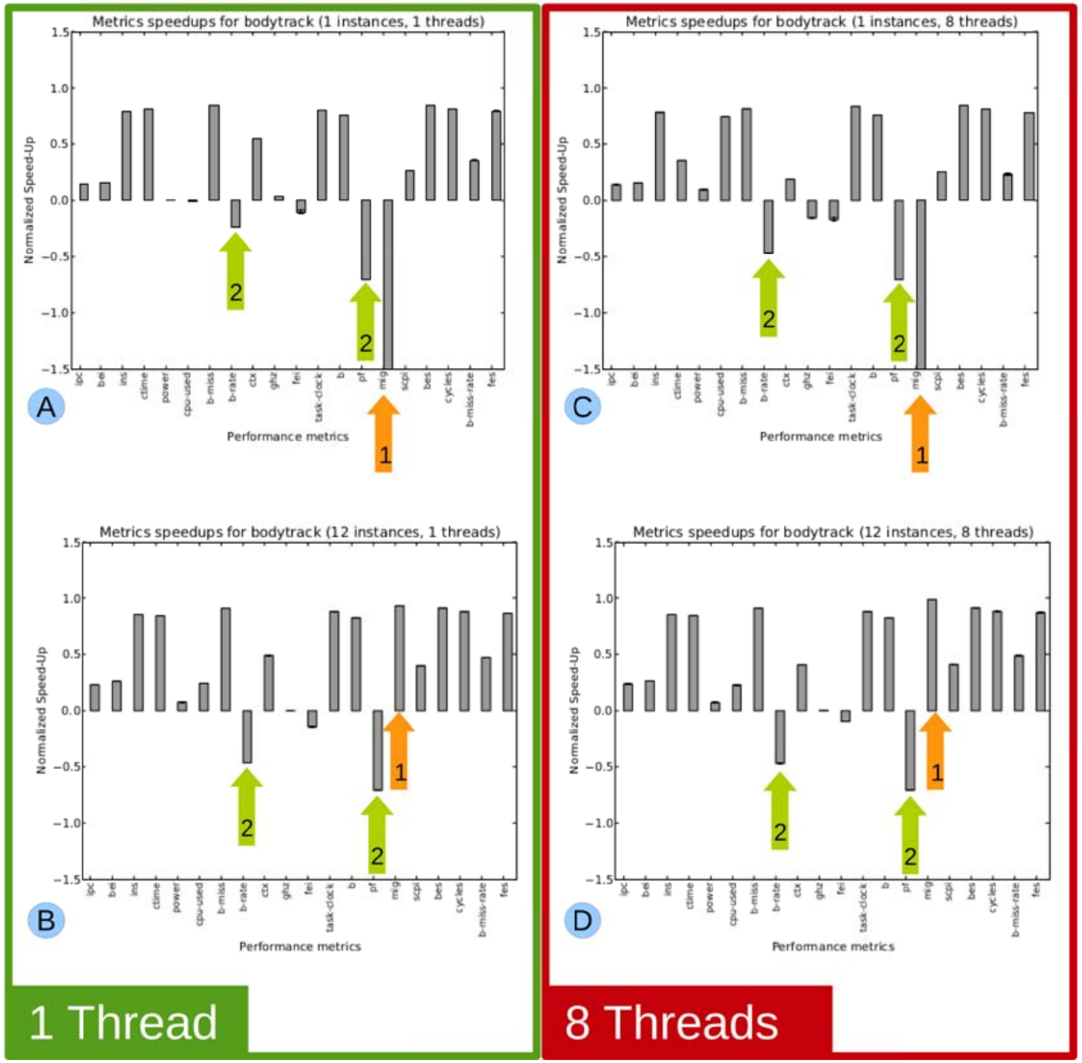
Improvements [%] - BBQ Manged vs Unmanaged

	Scenario	ctime [%Δ]	power [%Δ]	energy [%Δ]
A	1 Thread - 1 Instance	80	0.2	16
B	1 Thread - 12 Instances	84	7.8	655
C	8 Thread - 1 Instance	35	9.7	339
D	8 Thread - 12 Instances	84	7.2	604

Statistics based on: 30 runs, 99% confidence interval

Results
Benefits and Loss Comparison

Normalized speedups for all collected performance counters



positive bar corresponds to an improvement while a negative bar represent a deficiency of the managed application with respect of the original one

1 Same order of magnitude for "migrations" on lower congestions

2 "page faults" and "branch rate" always degraded because of code organization for BBQ integration

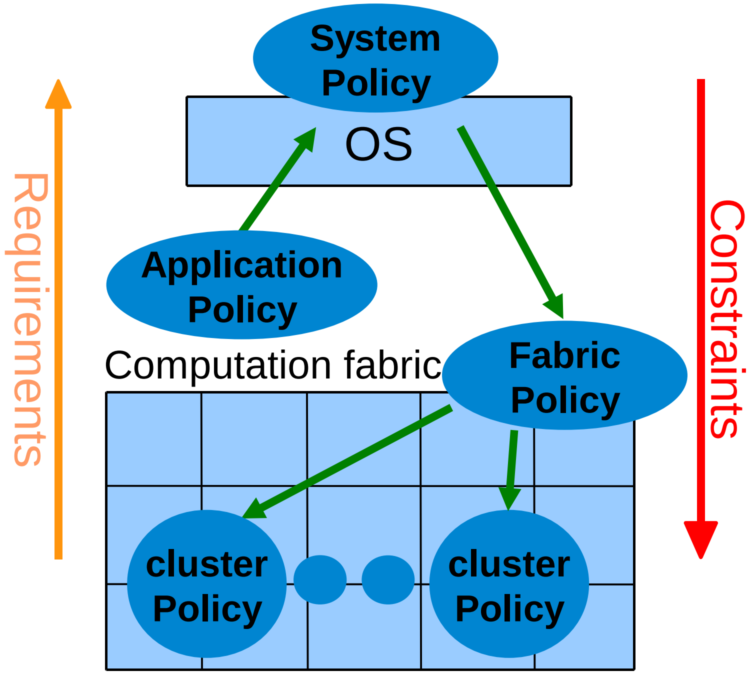
loop-unrolling could not be applied, but...
an improved integration has already been identified

Instruction stream optimization could be achieved by treading compile time optimization with effective resources assignment

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

Different granularity

- accellerated application
- operating system
- computation fabric
- computation clusters



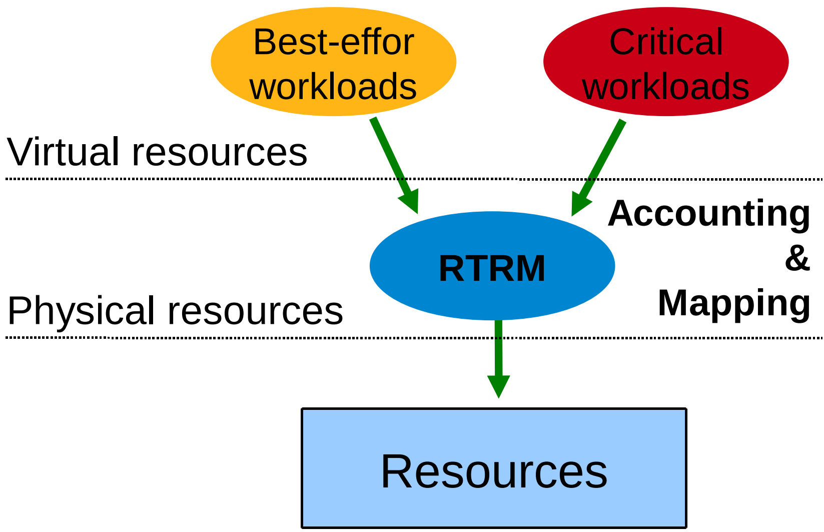
How to reduce control complexity?

Each granularity level collects requirements from lower levels 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



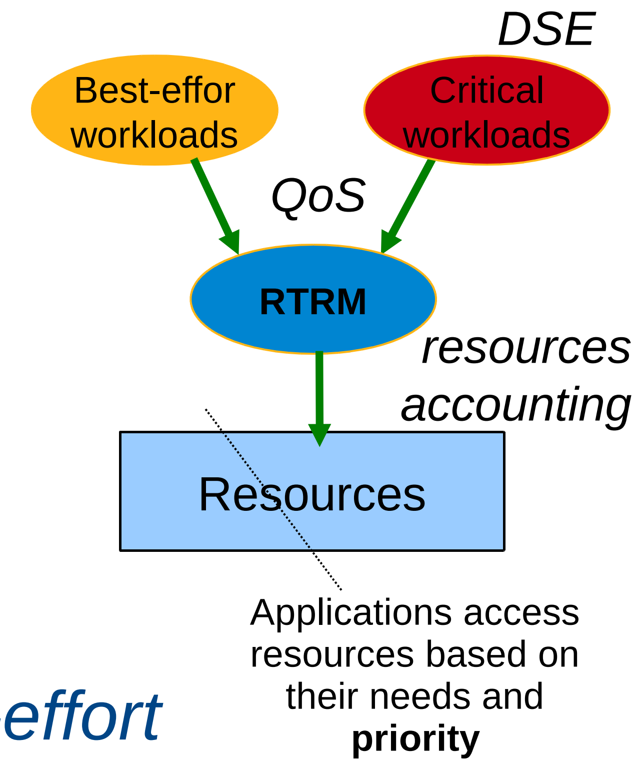
How to support optimal system resource exploitation?

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

The BarbequeRTRM Framework

Why such a name?!?

Because of its "sweet analogy" with something everyone knows...

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