HyVM – Hybrid Virtual Machines

Ada Gavrilovska
Karsten Schwan
Sudha Yalamanchili

and many PhD students

Project Context

Future HPC applications on heterogeneous platforms

- off (e.g., new NSF Track II machine)-chip and next, toward on-chip heterogeneous cores
- rich memory hierarchies, e.g., NUMA

New Execution Models

- diverse cores
- custom OS kernels
- I/O support

Accelerators and tool chains

- CUDA, OpenCL, …
- opportunity for compiler-based optimization methods

Future Applications

Media and image processing:

- for dynamic web content
- '**Snapfish**-like' imaging suite

Science and gaming:

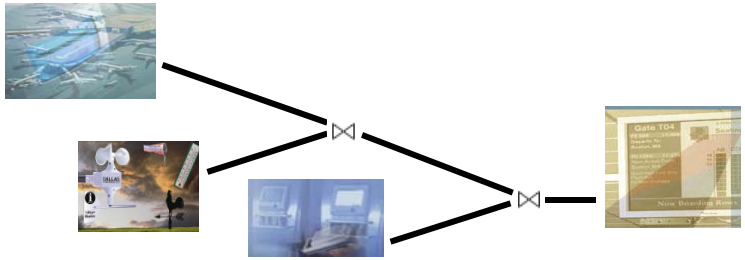
- Fusion modeling
- High perf. I/O: GTC, Pixie3D, XGC
- Libraries: **LAPACK, BLAS, VSIPL**, …

Financial and risk analysis:

- Black Scholes
- **Risk Analysis**
- **Derivatives processing**

Data-intensive and web:

- **Critical enterprise codes**
- Data mining
- Sensor data processing



Heterogeneous Platforms

- **Asymmetries**:
  - Performance
    - Different clock speeds, duty cycles
    - Differing issue widths
    - Varying cache sizes
  - NUMA memory
  - Toward ‘islands of cores’

Functional differences:

  - Diverse accelerators: heterogeneous:
    - Cell, GPU, Communications, Encryption, …
  - Shared ISA: asymmetric:
    - Missing SSE version
    - Missing floating point
    - Additional instructions for acceleration (Larrabee, …)

HyVM Project Goal

Uniform runtime model for heterogeneous platforms: Hybrid Virtual Machines

- **Uniformity**: software-based platform extensions:
  - ***Virtual Execution Unit*** *(VEU):* uniform runtime representation for program executables, targeting heterogeneous cores
  - ***Heterogeneity-awareness****:* system-level management methods for improved platform utilization (incl. cache and energy) and application performance (SLAs)
  - ***Dynamic platform emulation****:* runtime CK compilation or re-writing for diverse accelerator targets (via LLVM)
- **High performance:** diverse executables:
  - ***Commodity and custom VEU 'containers***': Virtual Machines (VMs) – processes/threads – commodity cores; Special Execution Environments (e.g., NVIDIA) - Computational Kernels (CKs) - accelerators
  - ***Runtime*** and **adaptive *{CK} optimization*** for parallelism
  - Standards-compliant CK programming and runtime APIs (OpenCL, CUDA)
  - Compiler-based optimization techniques for {CK}

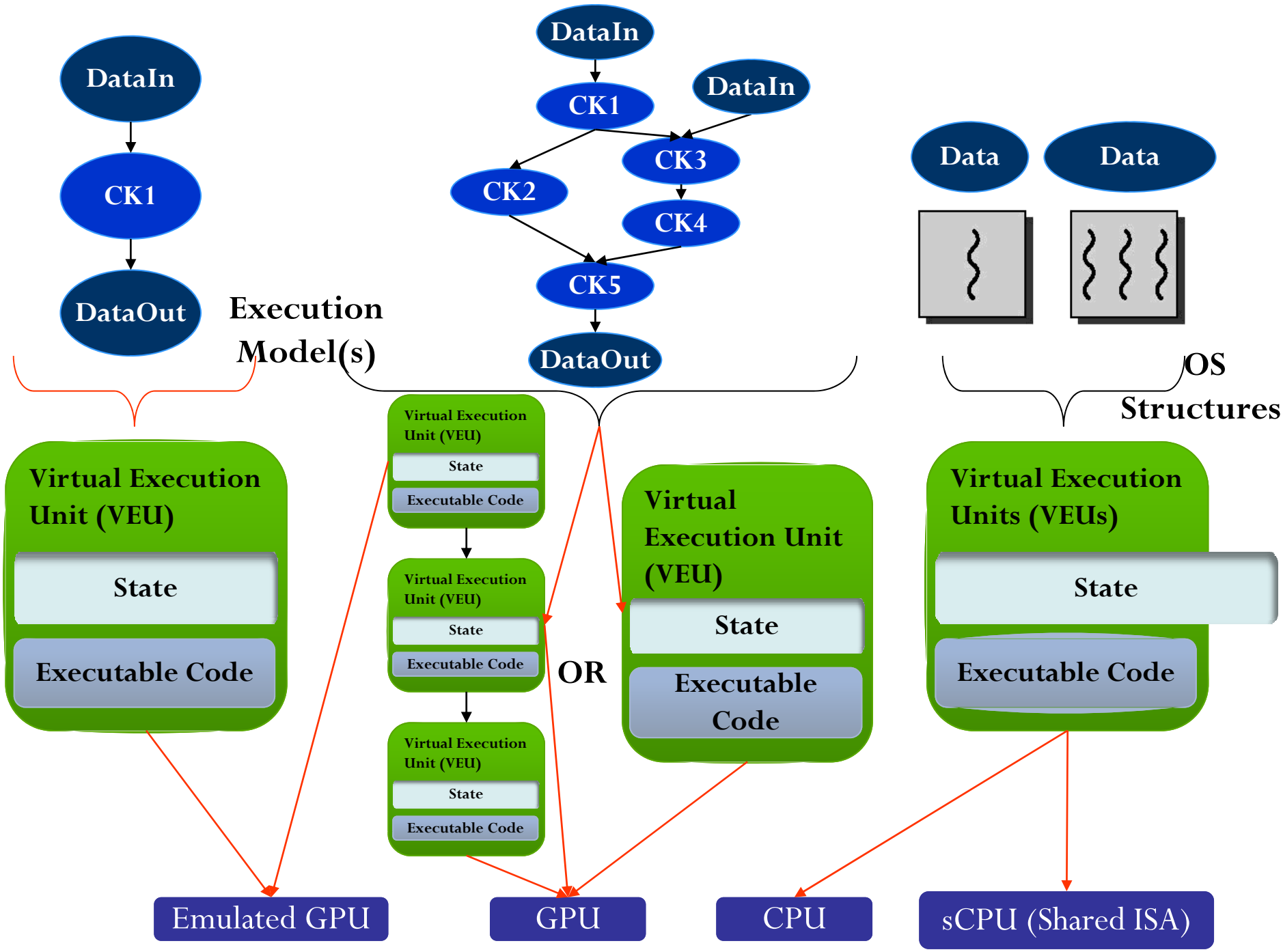
HyVM Project Elements

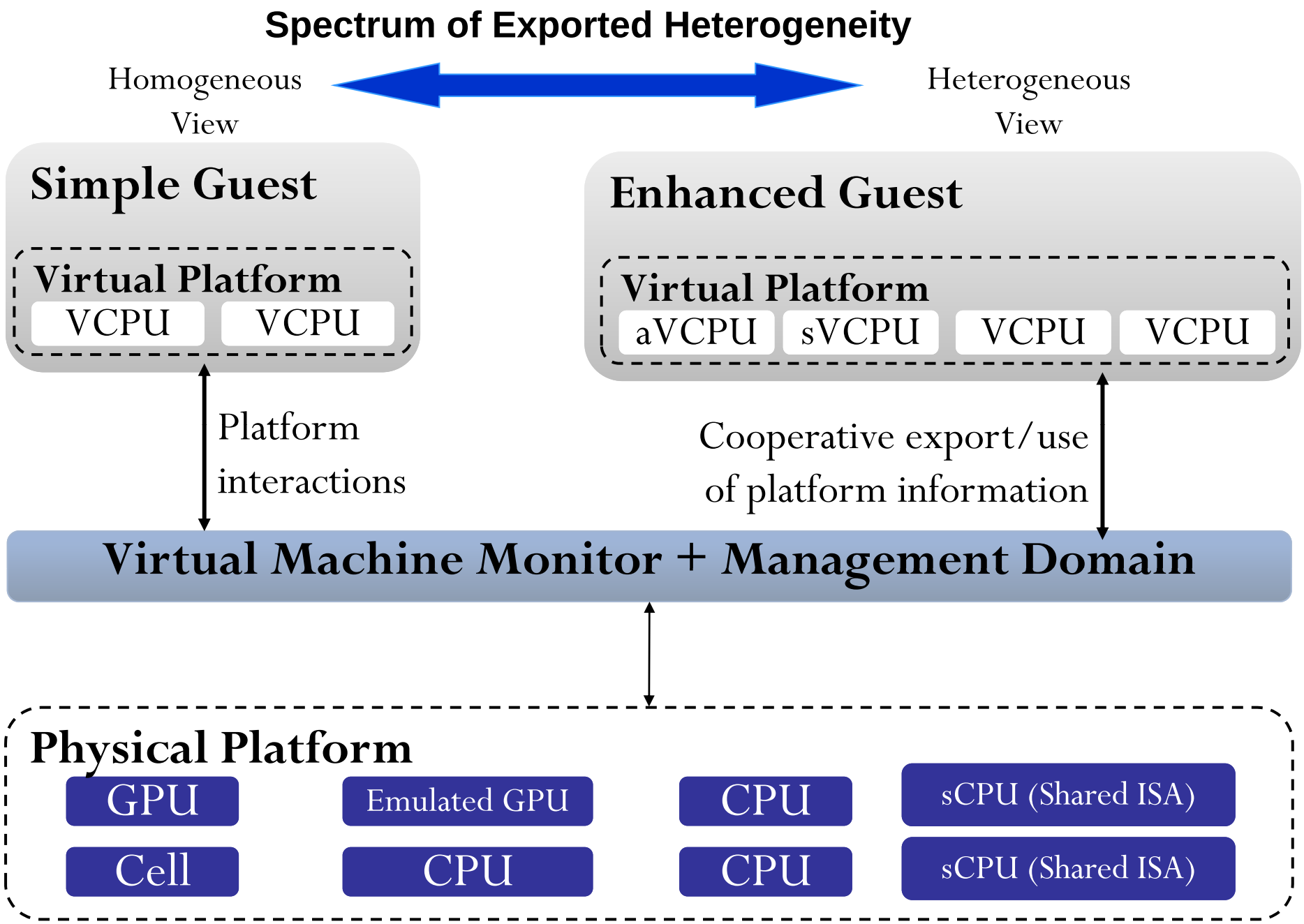
Attaining the uniform HyVM execution model

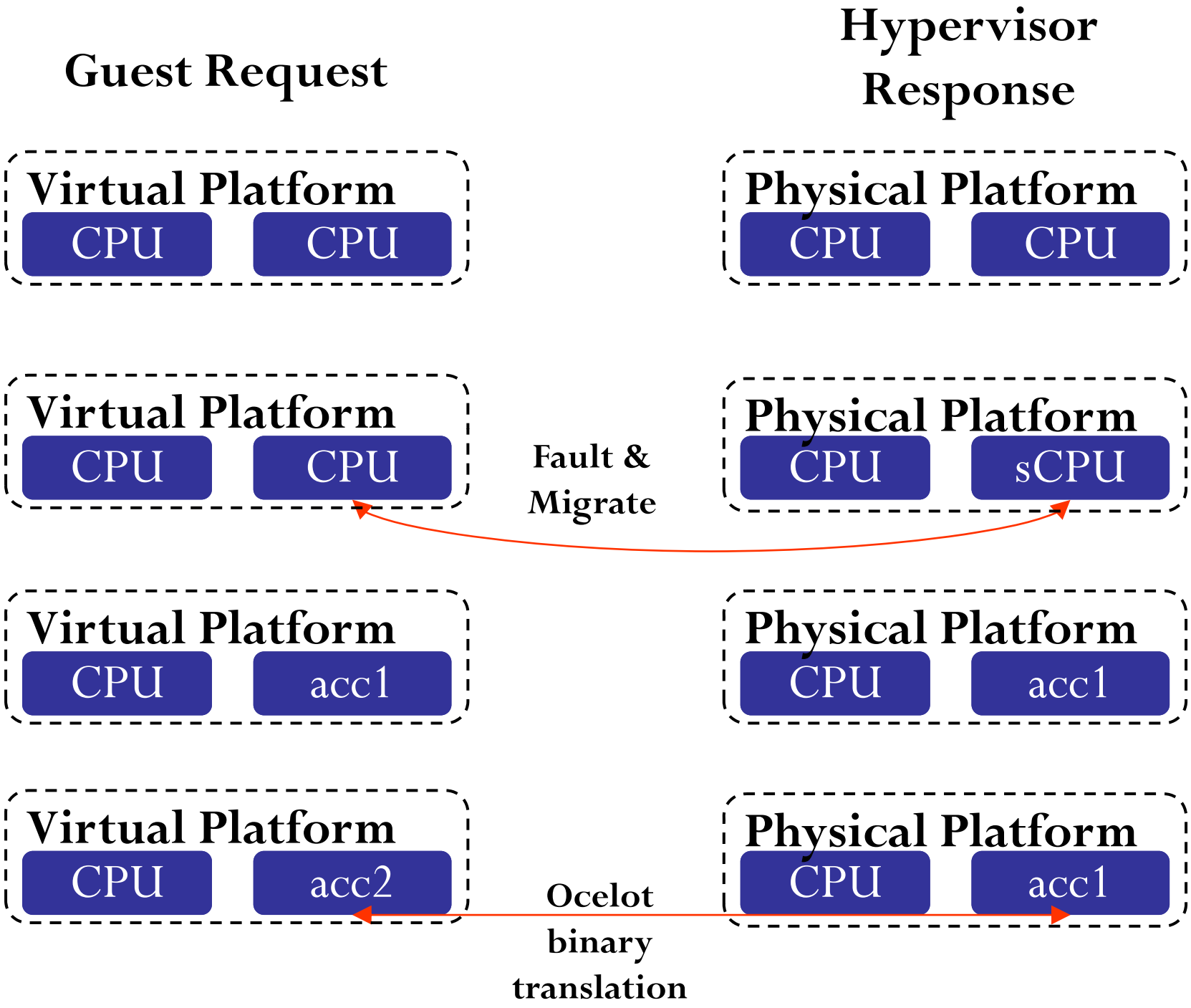
- **Leveraging virtualization technologies:**
  - Virtual Execution Units (VEUs)
    - Finer grain schedulable entities than VCPUs
  - Specialized execution environments (SEEs) for accelerators
    - **GViM** for efficient GPU virtualization (Niraj Tolia, HP)
    - **Cellule:** Dilma Silva, Jimi Xenides, Hubertus Franke (IBM)
  - **Montage**: dynamic resource management for sets of VEUs (SLA-awareness, runtime monitoring)
    - **Coordinated scheduling for accelerators** – **VMaCS**
    - **InTuneS** - system abstractions for coordinated management
    - **Cache-aware 'region' scheduling + correlation scheduling** for shared ISA VEUs – addressing NUMA and asymmetric platforms
  - Future hypervisors (using Xen): **heterogeneity-aware**
    - Extended work: scalable hypervisor structures

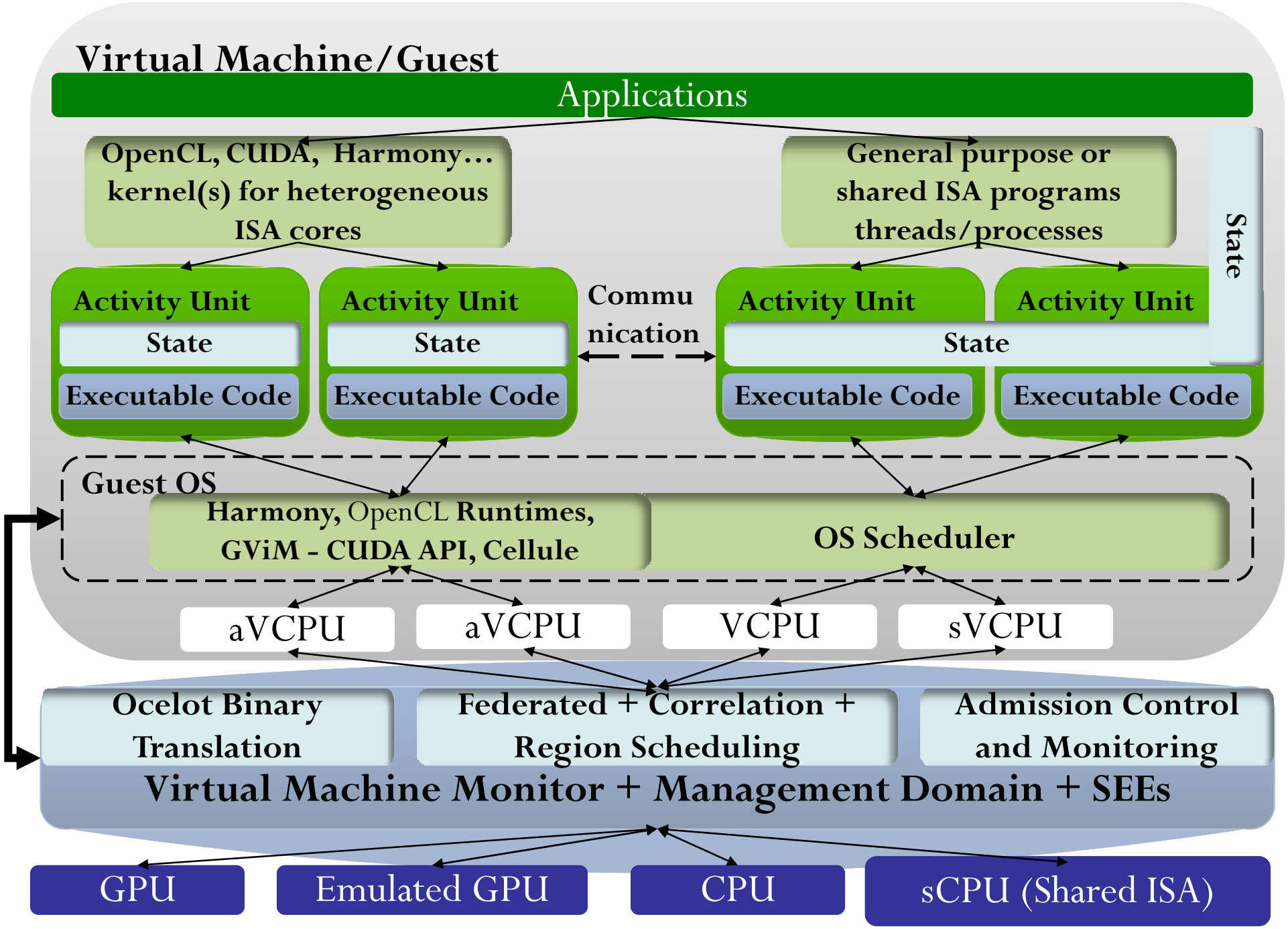
HyVM Project Elements

- **Leveraging evolving industry standards for accelerator APIs and interactions:**
  - Tool chain support for CUDA => LLVM/PTX translation, with future work on OpenCL – **Ocelot**
  - Exploiting compiler methods to optimize execution regimes for sets of VEUs – **Harmony**
  - Runtime APIs at CUDA level - **GViM**
- **Exploring new programming and execution models:**
  - Datalog to GPU compilation
  - Scheduling via on-line performance models and dynamic code generation for GPU vs. Host ISAs → Ocelot/LLVM
  - New analysis tools for debugging and performance tuning via emulation → Ocelot++


DataIn
CK1
DataOut
Execution Model(s)
DataIn
CK1
DataIn
CK3
CK2
CK4
CK5
DataOut
Data
Data
OS Structures
Virtual Execution Unit (VEU)
State
Executable Code
Virtual Execution Unit (VEU)
State
Executable Code
Virtual Execution Unit (VEU)
State
Executable Code
Virtual Execution Unit (VEU)
State
Executable Code
OR
Virtual Execution Units (VEUs)
State
Executable Code
Emulated GPU
GPU
CPU
sCPU (Shared ISA)


Spectrum of Exported Heterogeneity
Homogeneous View
Heterogeneous View
Simple Guest
Virtual Platform
VCPU
VCPU
Enhanced Guest
Virtual Platform
aVCPU
sVCPU
VCPU
VCPU
Platform interactions
Cooperative export/use of platform information
Virtual Machine Monitor + Management Domain
Physical Platform
GPU
Emulated GPU
CPU
sCPU (Shared ISA)
Cell
CPU
CPU
sCPU (Shared ISA)

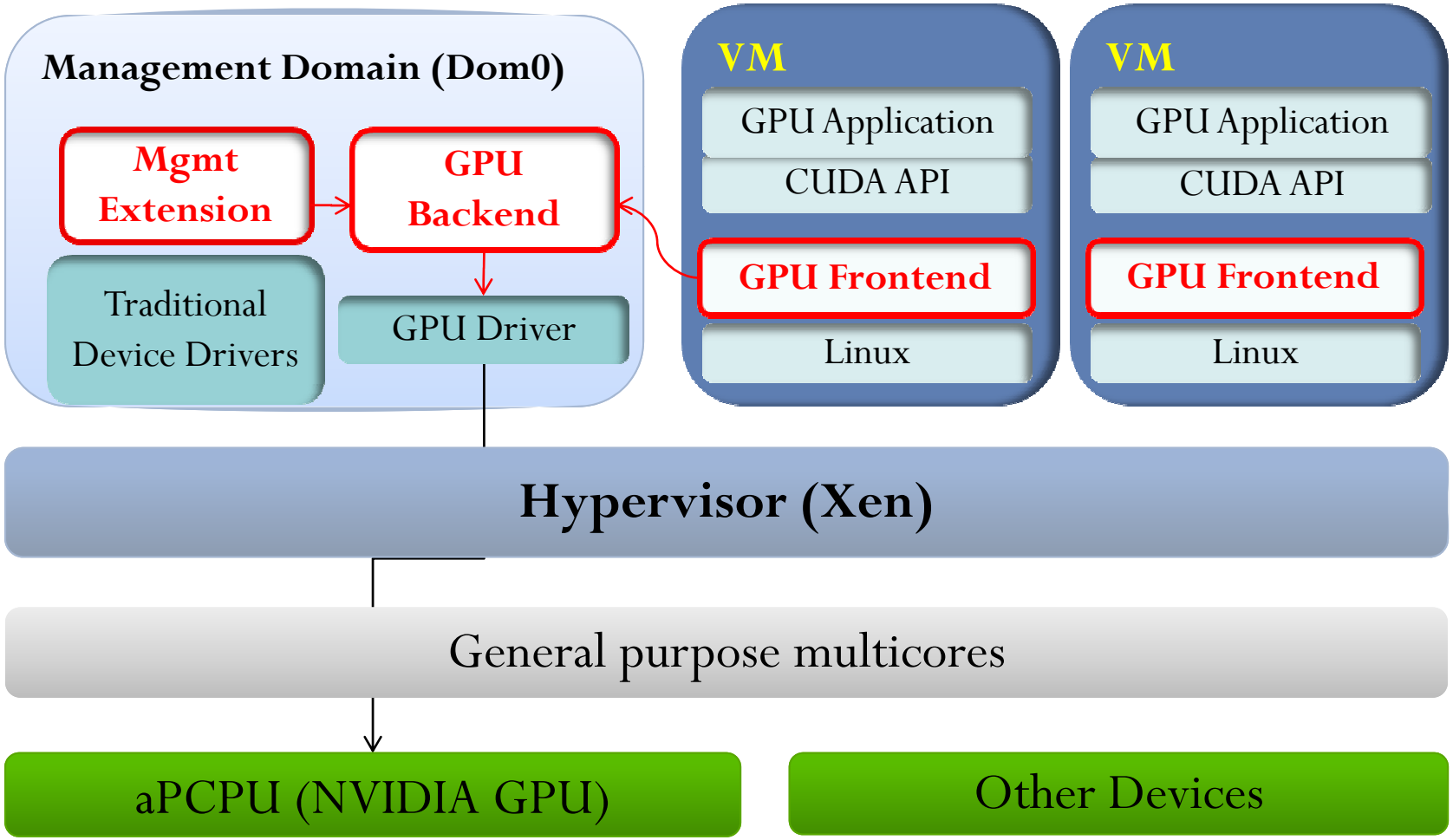

Guest Request
Hypervisor Response
Virtual Platform
CPU
CPU
Physical Platform
CPU
CPU
Virtual Platform
CPU
CPU
Fault & Migrate
Physical Platform
CPU
sCPU
Virtual Platform
CPU
acc1
Physical Platform
CPU
acc1
Virtual Platform
CPU
acc2
Ocelot binary translation
Physical Platform
CPU
acc1


Virtual Machine/Guest
Applications
OpenCL, CUDA, Harmony… kernel(s) for heterogeneous ISA cores
General purpose or shared ISA programs threads/processes
State
Activity Unit
State
Executable Code
Activity Unit
State
Executable Code
Commu nication
Activity Unit
Activity Unit
State
Executable Code
Executable Code
Guest OS
Harmony, OpenCL Runtimes, GViM - CUDA API, Cellule
OS Scheduler
aVCPU
aVCPU
VCPU
sVCPU
Ocelot Binary Translation
Federated + Correlation + Region Scheduling
Admission Control and Monitoring
Virtual Machine Monitor + Management Domain + SEEs
GPU
Emulated GPU
CPU
sCPU (Shared ISA)

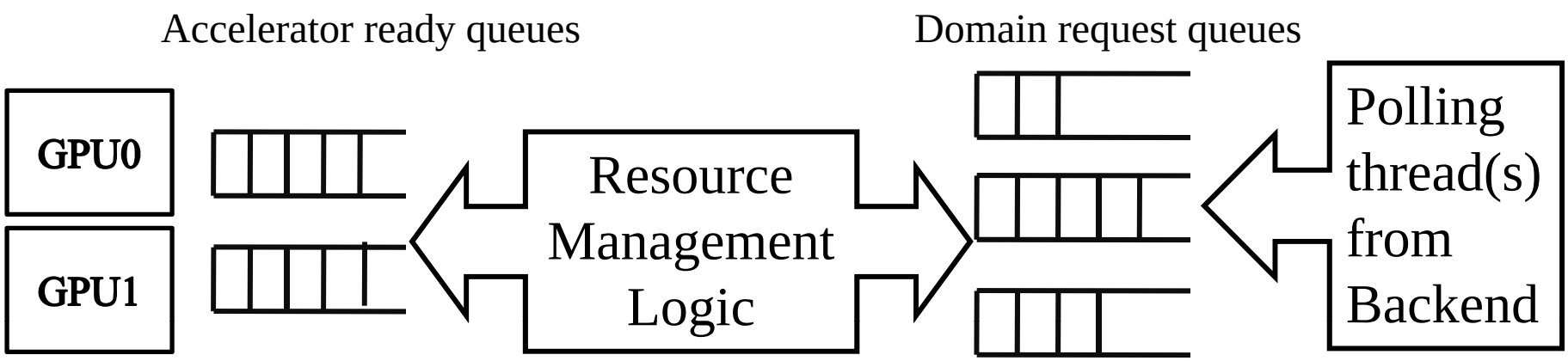
Technical Elements – GViM and VMaCS

Vishakha Gupta, Niraj Tolia, others

Extending Xen for efficient {CK} execution



Technical Elements – GViM + VMaCS

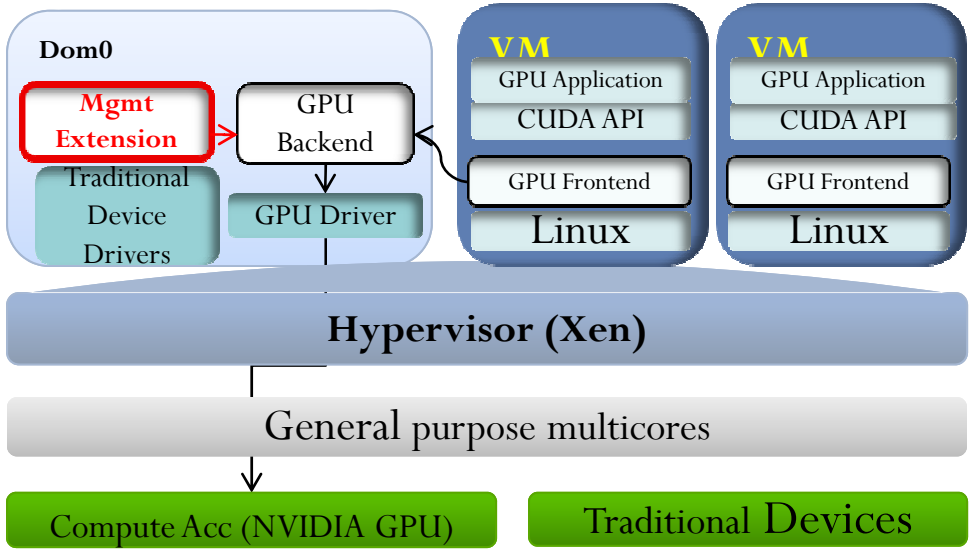


GViM:

- **VEUs on gVCPUs**

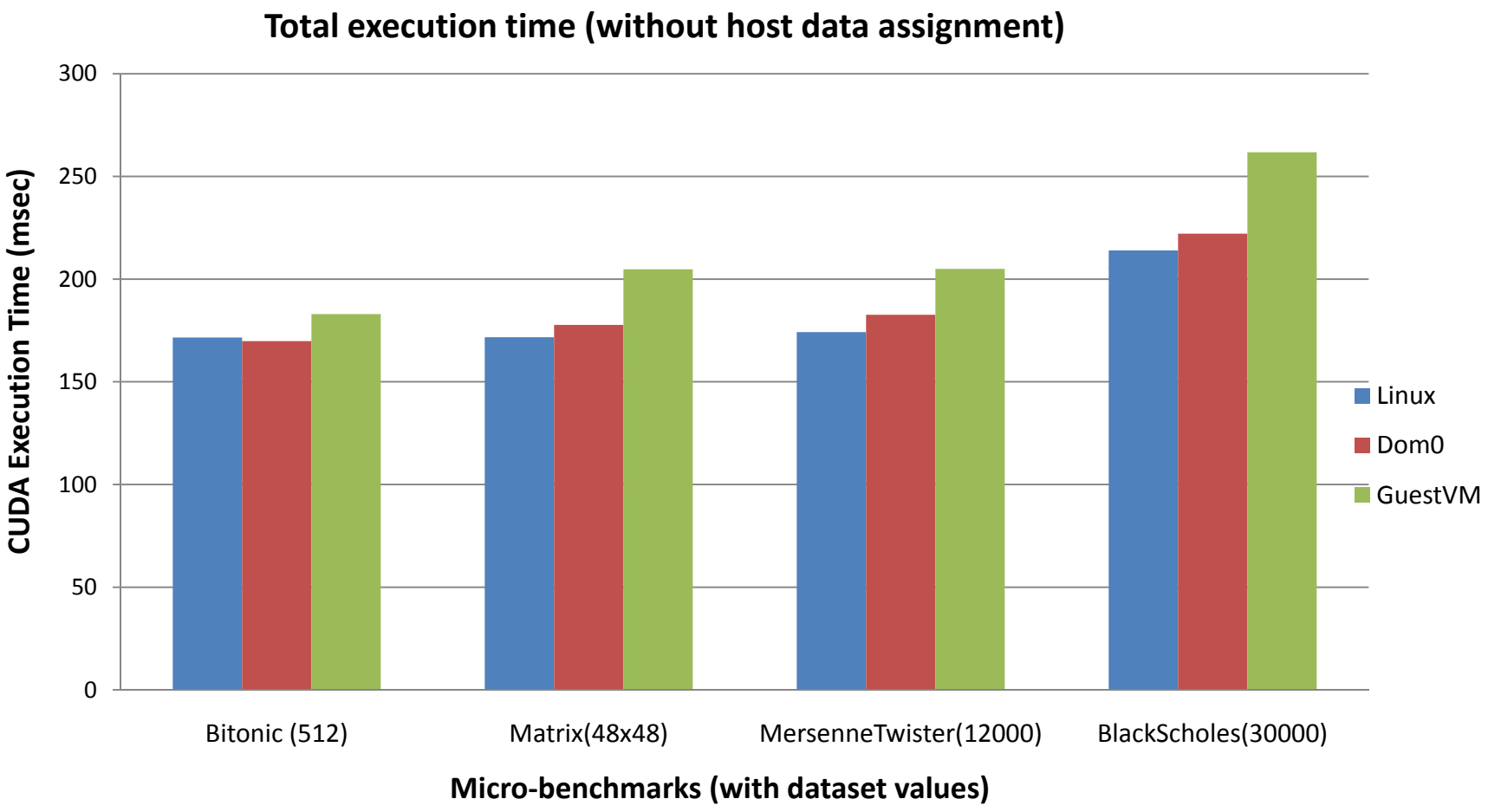
VMAcS:

- **Management Extensions for *Coordinated Scheduling***



Technical Elements – GViM

GPU Guest Performance – Baseline Results



Less than 15% slowdown in the worst case
Further improvements made/underway

VMaCS Scheduling Methods - Examples

- Round robin (RR)
  - Some guest domain chosen every period
  - Poller monitors its queues for the period
  - Moves to the next domain
- XenoCredit
  - Guest credits assigned at boot time
  - Use these credits to calculate proportions
  - Poll domains for time proportional to their credits

BlackScholes: XenoCredit vs. Round Robin

- BlackScholes with 60000 options and 4096 iterations
- Host with 2 GPUs and 4 guest VMs

VMs	Credits	Expected iter/ms	RR	XC
VM1	512	1.8	2.632	1.867
VM2	256	2.4	2.567	3.12
VM3	256	2.2	2.804	3.213
VM4	256	2.54	2.901	3.45

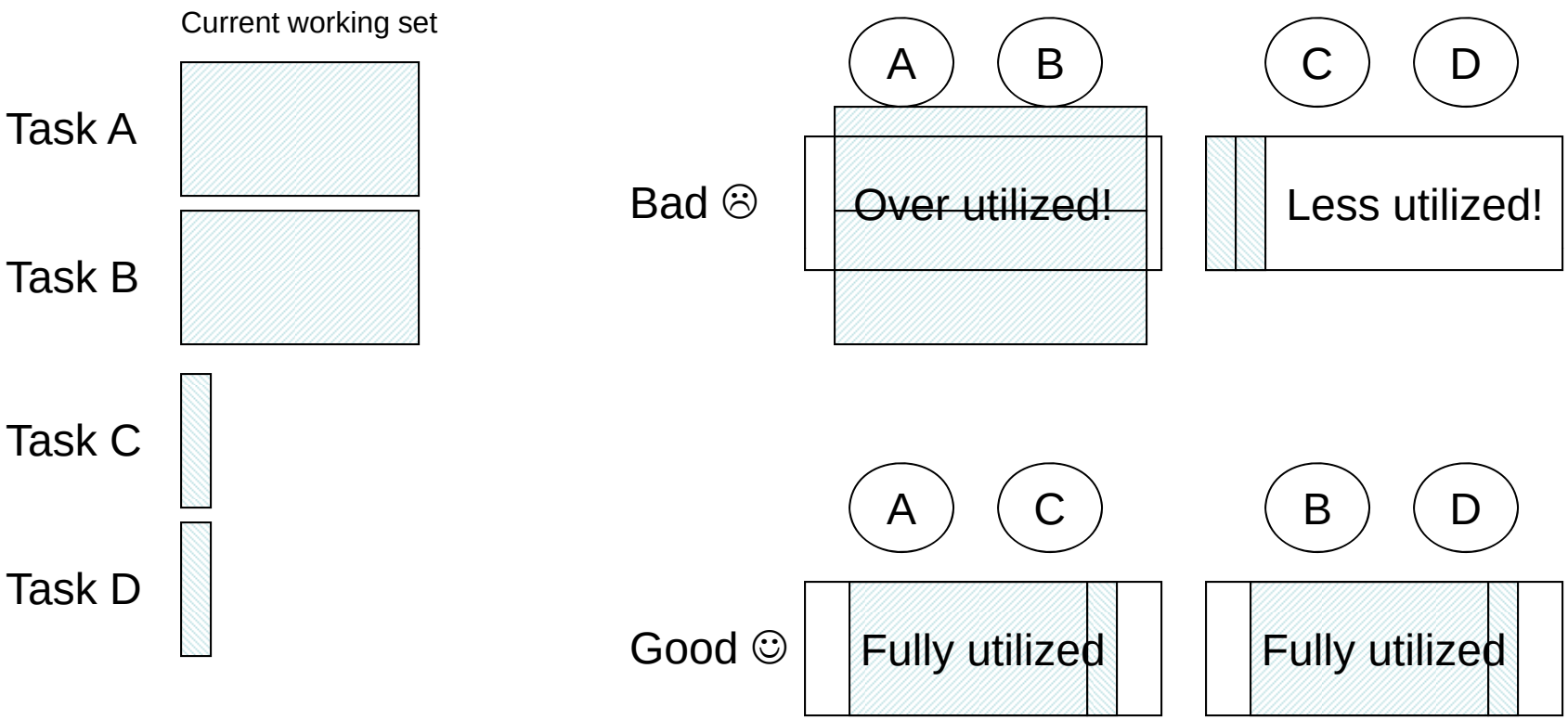
Motivates new methods for resource management in accelerator-based systems – now being completed

Technical Elements - Region and Correlation Scheduling

Min Lee, Vishakha Gupta, Rob Knauerhase

- Targeting Asymmetric Multicore Platforms
- Using Intel quad core and Nehalem server architectures
- Region scheduling – dealing with the NUCA=>NUMA nature of future manycore systems + NUMAFix (Dulloor Rao)
- Correlation scheduling – dealing with asymmetric codes with shared ISA

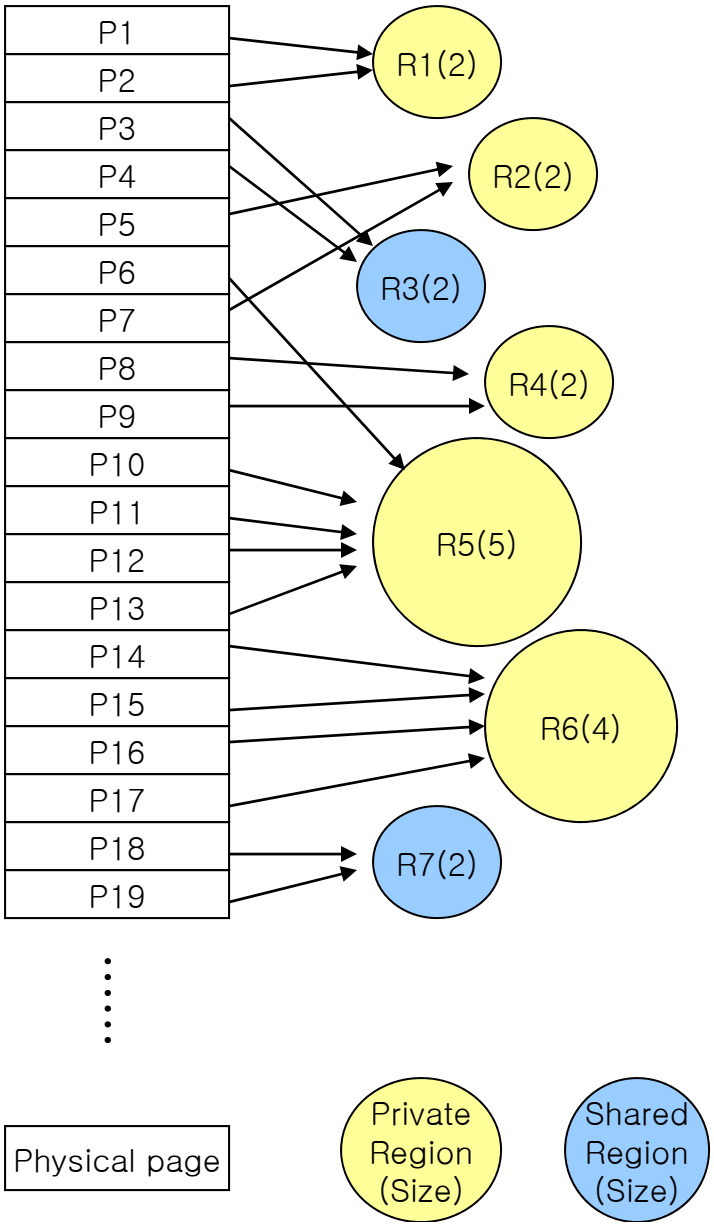
Region Scheduling:
Cache Aware Scheduling for CMPs



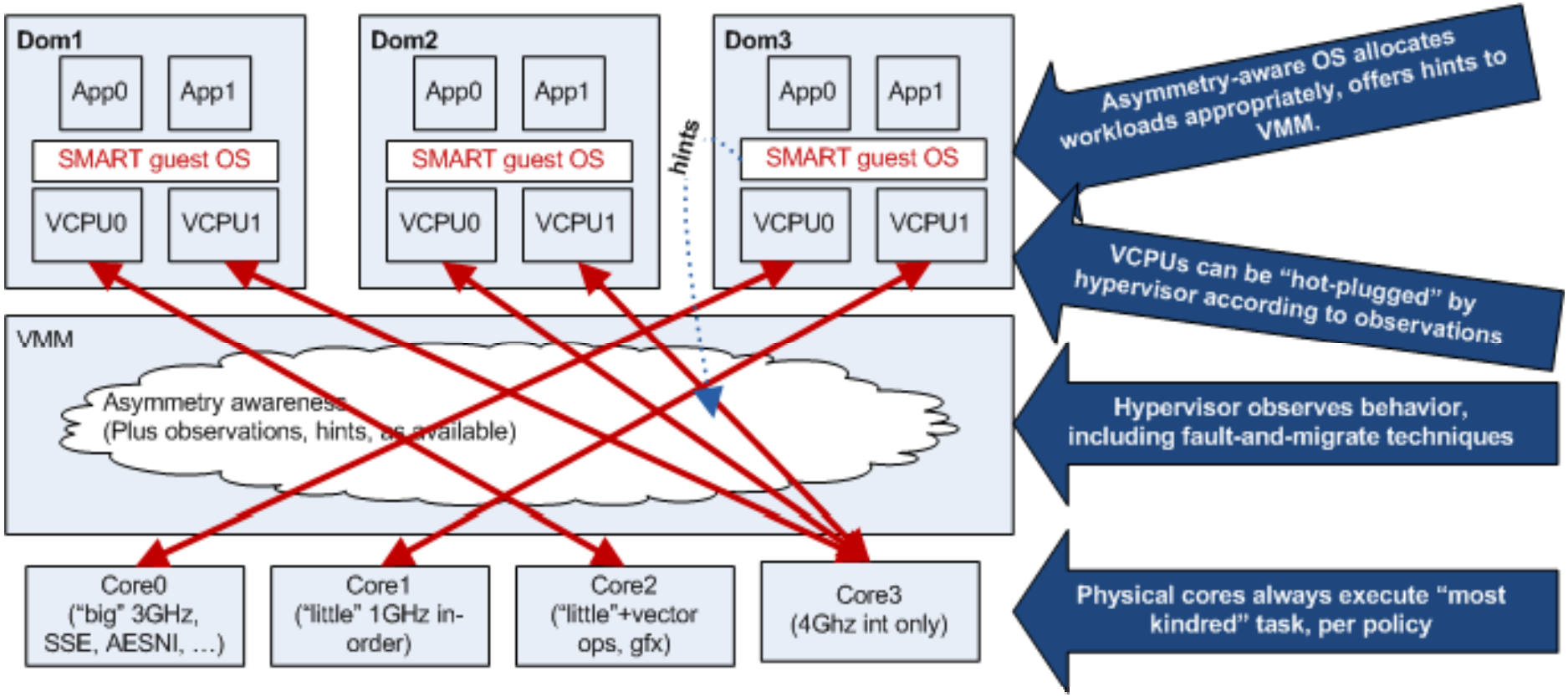
- Issues
  - Working set estimation – size, constituent elements (unique and shared)
- Approach
  - Runtime ‘region’ determination to capture these working set

Region Scheduling – Basic Idea

- Partition physical memory into regions
- ***Schedule regions on caches***
- Core can access only the allowed regions
  - System-level enforcement (using page tables)
- Private/Shared regions
  - we focus on private regions

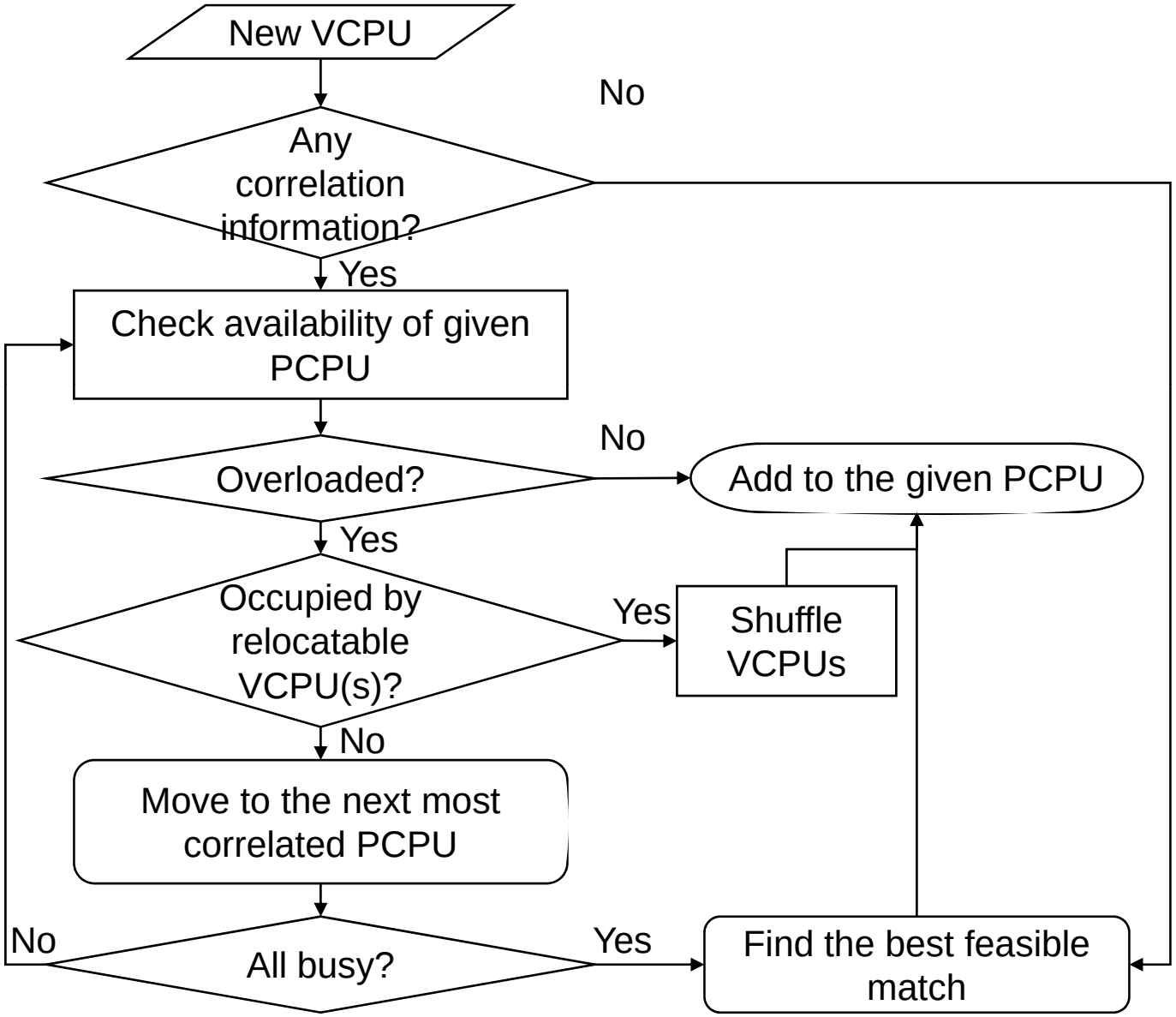


Correlation Scheduling: Mapping to Asymmetric Cores



Asymmetry-aware Hypervisor supports asymmetric cores and NUMA memory

VCPU Admission Flowchart



Technical Elements – ‘Islands of Cores’

Priyanka Tembey
Dulloor Rao
Mukil Kesavan

Motivation: many-core platforms

- Multiple execution and scheduling contexts: 'islands'
  - differ in capabilities, micro-architectures
  - have NUMA memory characteristics
- Inter-connection variants – current platforms
  - PCIe, shared caches, fast point-to-point Interconnect (QPI/Hypertransport)
- Inter-connects – future platforms
  - complex cache structures (!)
  - PGAS memory with integrated memory controllers/inter-connects?

Technical Elements – Islands of Cores

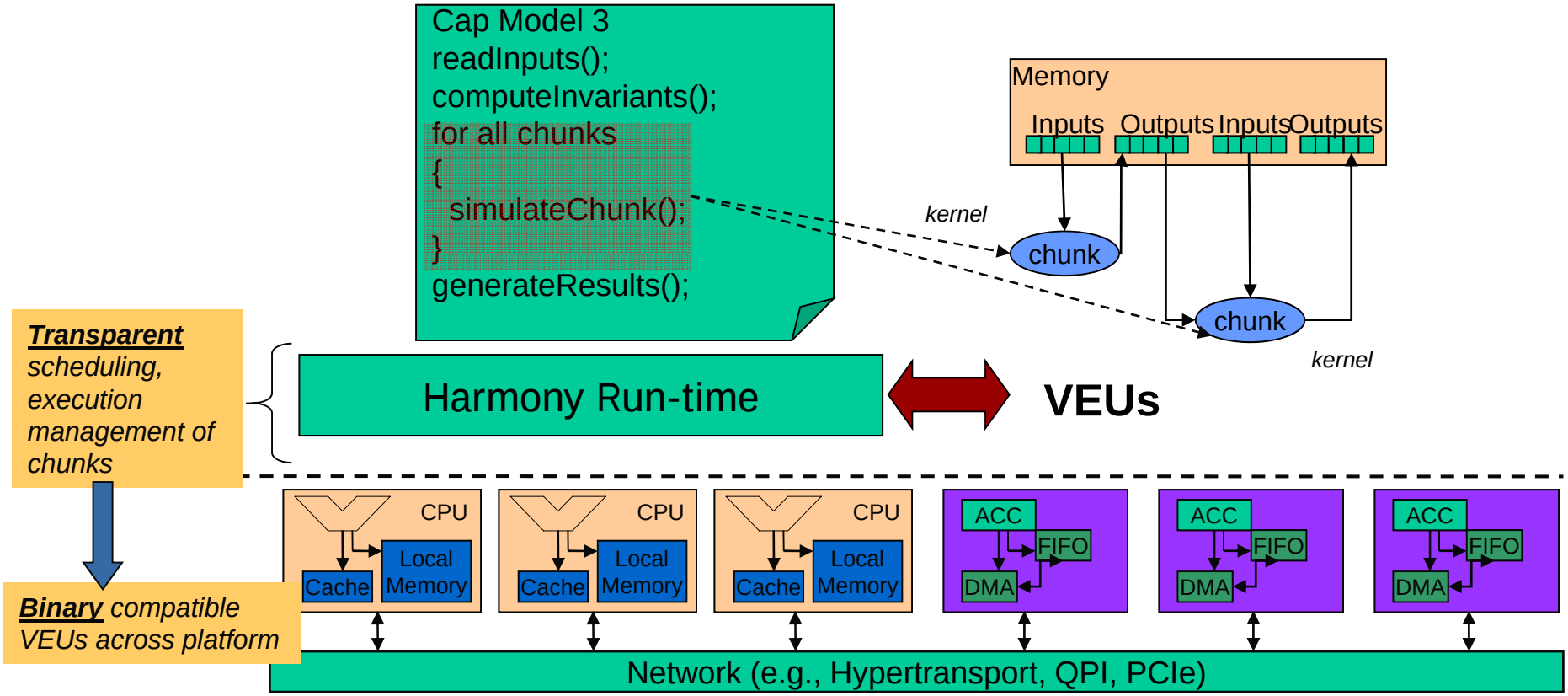
- InTuneS: Priyanka Tembey
  - Inter scheduler-island lightweight event channel
  - Generic yet expressive definition of inter-island message semantics
  - Gray-box scheduler coordination interface
    - scheduler implementation-agnostic API
  - Application/VM-scheduler interface for explicit SLA policy enforcement
  - Application/VM-monitoring for implicit observation-based policy enforcement
  - Evaluation with complex scheduler (Linux) and hypervisor scheduler (HV in progress), initially using communication accelerator

- Scalable hypervisor structures: Mukil Kesavan and Dulloor Rao
  - Hypervisor structure should match `islands of cores'
  - e.g., consider Xen `Dom0' bundling of functionality

  $\Rightarrow$ offer solutions for isolation with bundled functionality

  $\Rightarrow$ Find ways to efficiently run many `small' domains, specialized, each with different tasks

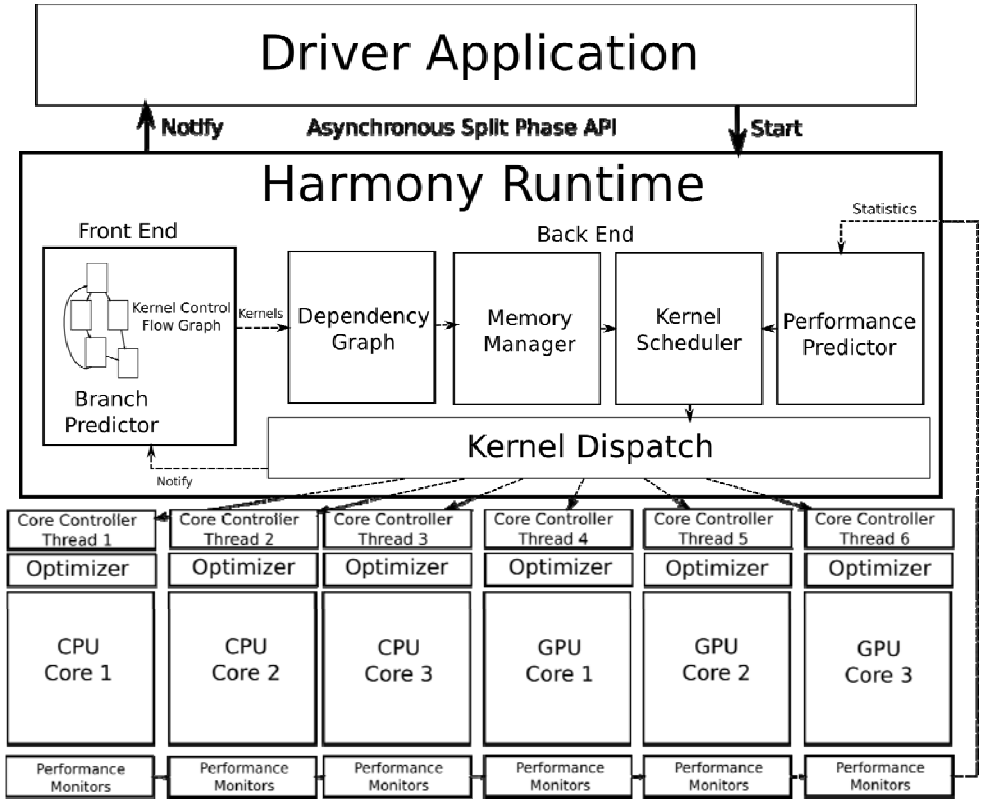
Technical Elements – Creating, Optimizing, and Re-targeting VEUs via Harmony and Ocelot - Gregory Diamos, Andrew Kerr



- Minimize/avoid re-tuning and porting applications as you add accelerators
- Advanced optimizations
  - Speculation, performance prediction, kernel fusion

Scalable Portable Execution – Harmony Runtime

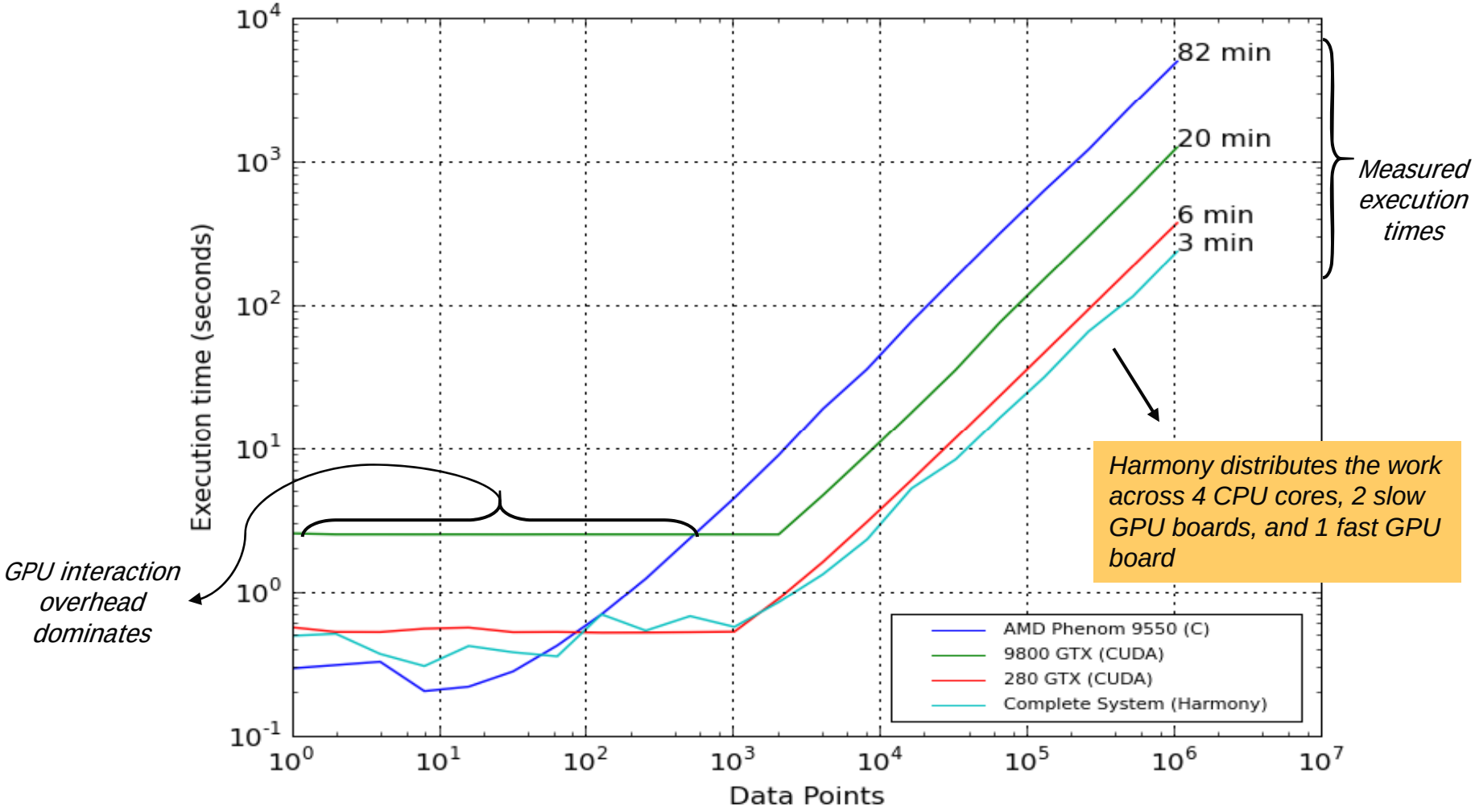
Current Single Node/Multi-GPU Version



Future extensions of Harmony functionality will be to operate across multiple nodes. Intent:

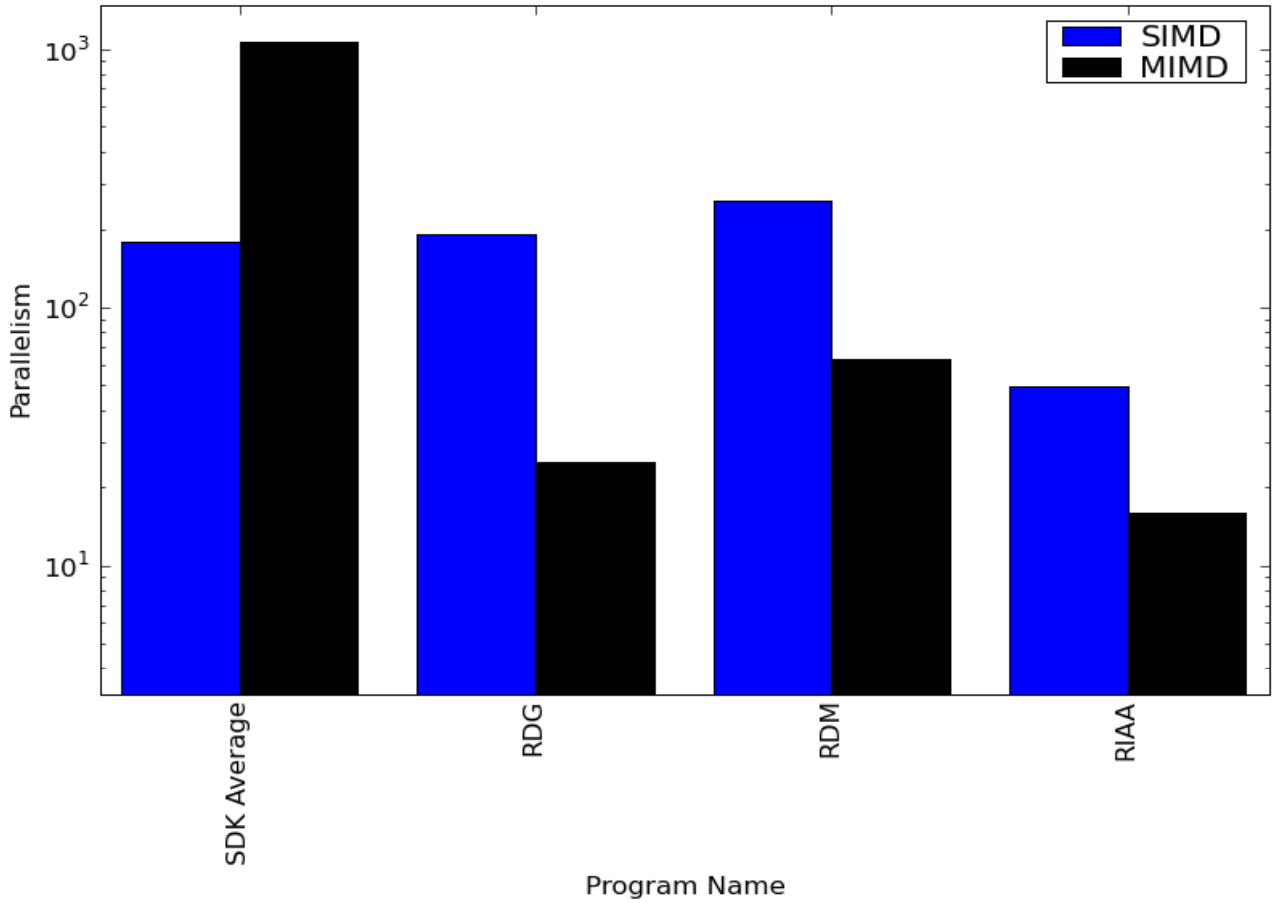
1. Decision models for remote vs. local execution
2. Transparent deployment and execution of GPU kernels on remote GPUs
3. Inclusion of performance optimizations, including:
   - i) speculative kernel execution,
   - ii) data movement, and
   - iii) memory footprint vs. concurrency tradeoffs

Problem Scaling – Risk Analysis Application



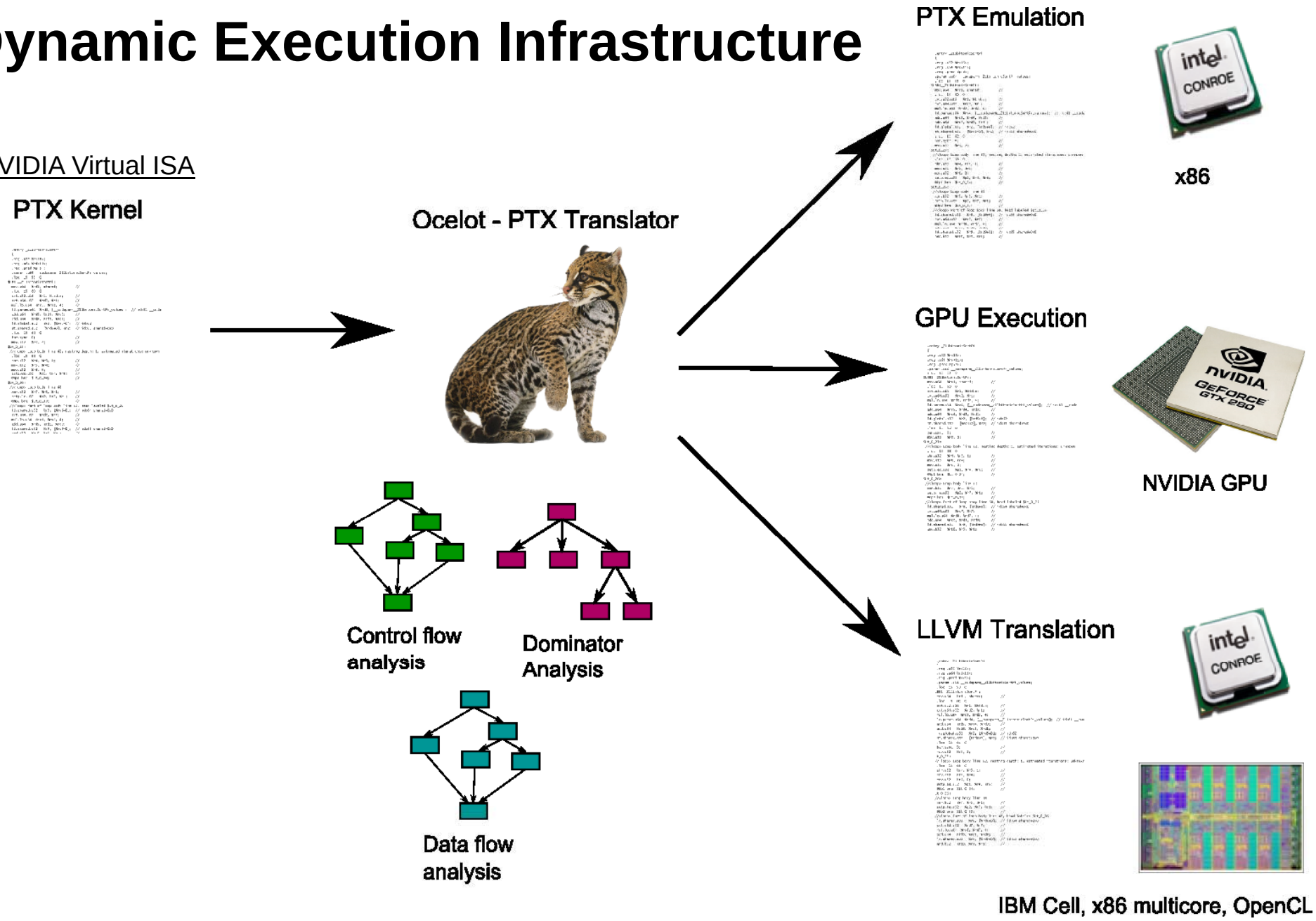
Scaling of GPGPU Applications

*Average applications can scale to **1015** cores, each with **180** element -wide SIMD units*

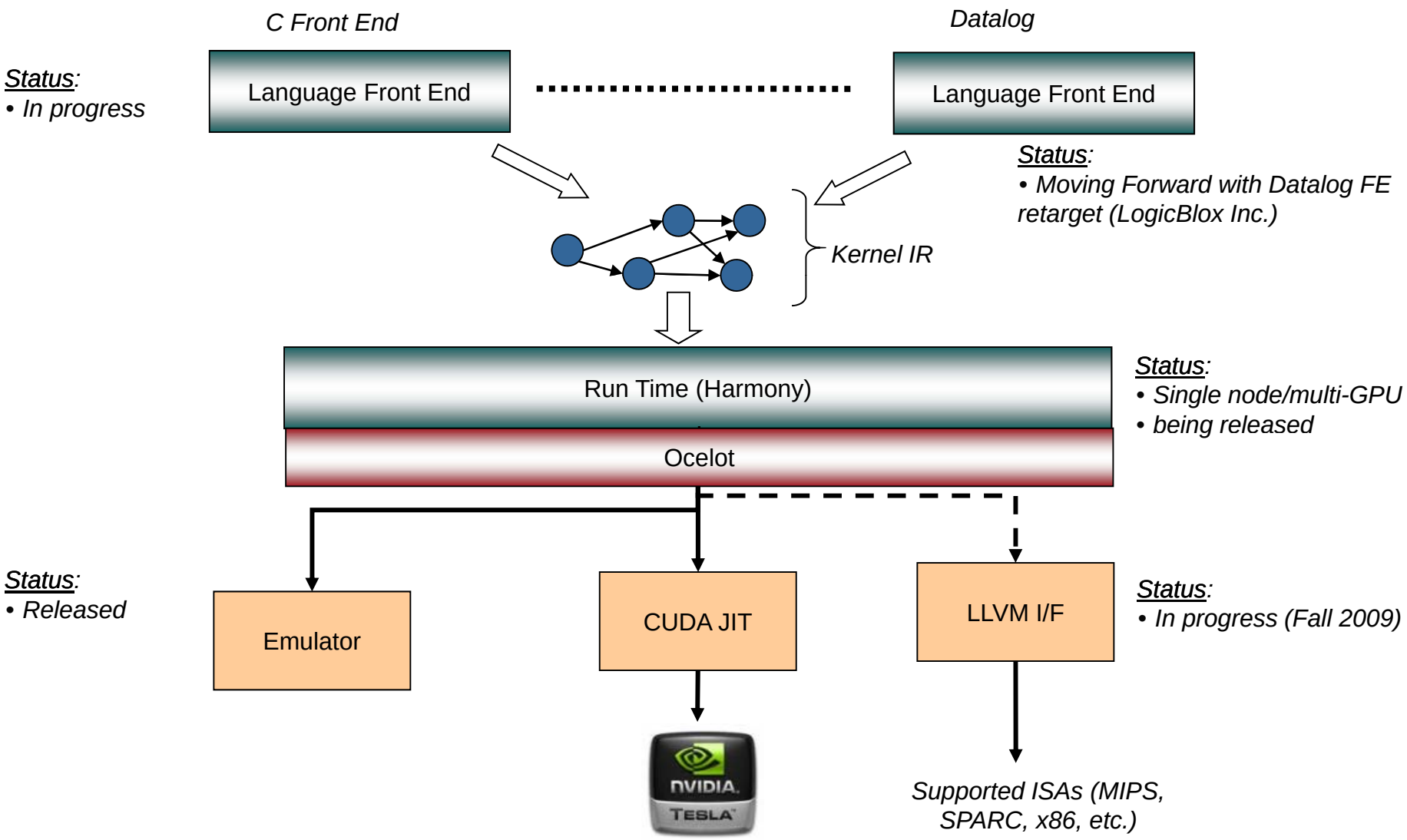


Using the Ocelot CUDA Emulator

Ocelot:
Dynamic Execution Infrastructure



Emerging Environment



Future Directions

- **Why HyVMs**: future applications:
  - Media-rich web applications
    - online `photoshop' (with HP)
    - dynamic stream customization (with Motorola)
  - Financial and HPC codes
    - Benchmarks (e.g., Black Scholes – IBM, HPC – NVIDIA, Interactive - Intel)
    - Petascale codes (with DOE)
    - Petascale I/O (with DOE)

- **From platforms to clusters to large-scale data centers:**
  - Distributed execution model for petascale machines
  - Other execution models: data-intensive (Hadoop, System S, …)

- **Future platforms:**
  - **Power/performance** tradeoffs and implications
  - Memory hierarchies and their effects (on- vs. off-chip)
  - Alternative memory models – e.g., PGAS
  - Massively parallel HyVMs – e.g., 1000s of lightweight domains
  - Tool chains: future standards (e.g., OpenCL), compiler research (other faculty)

Broader Context
CERCS Research Center

- **CERCS – Center for Experimental Research in Computer Systems (www.cercs.gatech.edu)**
  - **NSF Center**
  - **Member companies include HP, IBM, Intel, and Atlanta companies like Logicblox and ICE, plus several others**
  - **ORNL key partners: Steve Poole, Jeff Nichols, Barney Maccabe, Scott Klasky**

- **Opportunities for large-scale experimentation:**
  - New award for large-scale hybrid cluster machine: **NSF Track II award**, Vetter, PI, Schwan and Yalamanchili Co-PIs, first machine to be built Winter 2010
    - provides additional funding to explore 'distributed execution environment' on high end machine and for 'distributed performance emulation of multi-GPU machines'

- **Project growth and impact through CERCS efforts and faculty:**
  - Involving other faculty:
    - Santosh Pande – Glimpses - finding accelerator functions in application codes (NSF)
    - Hyesoon Kim – Prospector – precise performance (and power) estimation for next generation accelerators (with additional support from Intel – Larrabee – NVIDIA and NSF) – IBM ArchPIC - Valentina Salapura
    - Nate Clark – program analysis (Google)
    - Matt Wolf – I/O acceleration for petascale machines (DOE ORNL)
  - Working with technology and application partners:
    - IBM: financial and HPC apps., focus on system-level esource management
    - NVIDIA: student fellowships – Harmony, Ocelot, equipment donations
    - Intel: exploring asymmetric architectures
    - HP: CUDA-based virtualization, coordinated scheduling, data center and web applications
    - Logicblox: accelerating Datalog applications used in retail management and insurance risk estimation
  - Additional research support:
    - NSF Track II award – Oct. 2009
    - NSF Modeling and Simulation Award (recommended) October 2009 <integrating Ocelot with QEMU>
    - NSF SHF program award – Sept. 2009
    - DOE and NSF: I/O for petascale machines, Jan. 2009
    - Steve Poole: Power/Performance tradeoffs
    - HP: coordinated scheduling and large-scale data center management, Aug. 2008
    - Intel: IA-compatible VMs for heterogeneous many-core systems, since 2007
  - Linkage to other projects:
    - The Energy Stack

Publications and Resources

Project web page: http://www.cercs.gatech.edu/projects/HyVM/

Recent Publications:

- Priyanka Tembey, Anish Bhatt, Dulloor Rao, Ada Gavrilovska, Karsten Schwan, “Flexible Classification on Heterogeneous Multicore Appliance Platforms”, ICCCN'08, Aug. 2008
- A. Kerr, G. Diamos, and S. Yalamanchili, "A Characterization and Analysis of PTX Kernels," to appear in the 2009 IEEE International Symposium on Workload Characterization, October 2009.
- V. Gupta, J. Xenidis, P. Tembey, K. Schwan, A. Gavrilovska, "Cellule: Lighweight Execution Environment for Virtualized Accelerators", poster session, SuperComputing'09, Nov. 2009.
- G. Diamos and. S. Yalamanchili, "Kernel Level Speculation”, under submission.
- V. Gupta, J. Xenidis, K. Schwan, "Cellule: Lighweight Execution Environment for Virtualized CellB.E.", under submission.
- Min Lee and Karsten Schwan, "Region Scheduling: Virtual Machine Scheduling for Next Generation Multicore Platforms", in preparation.
- Vishakha Gupta, Rob Knauerhase, and Karsten Schwan, “Asymmetric Hypervisors”, in preparation.
- Priyanka Tembey, Ada Gavrilovska, and Karsten Schwan, “InTuneS: System Abstractions for Coordinated Scheduling”, in preparation.

Software:

- Ocelot, can be obtained at http://code.google.com/p/gpuocelot/, including documentation.
- Harmony programming model: http://www.ece.gatech.edu/research/labs/casl/harmony/hpdc5hot-diamos.pdf