The Power of Many: Towards a Convergence of HPC-HTC Computing

Shantenu Jha

Rutgers Advanced Distributed Cyberinfrastructure & Applications Laboratory (RADICAL)

http://radical.rutgers.edu
Outline

- The case for Building Blocks for "Workflows"
  - A fresh perspective on workflows
  - RADICAL-Cybertools: Building blocks that support abstractions & models based approach to scalable and extensible workflows

- Computer Science Results:
  - RADICAL-Pilot design and implementation supports the efficient launch and management of $O(10K)$ tasks over 64K cores.

- Domain Science Results:
  - Better Sampling using ExTASY Workflows

- **Abstractions** and **execution models** unify HTC & HPC under the same conceptual framework and implementations.
  - Distinctions are meaningless and artifacts of software systems!
A Fresh Perspective on Workflows

- Initially “Monolithic” Workflow systems with “end-to-end” capabilities
  - Workflow systems were developed to support “big science” projects.
  - Software infrastructure was “fragile”, unreliable, missing services

- Workflows aren’t what they used to be!
  - More pervasive, sophisticated but no longer confined to “big science”
  - Diverse “design points”; unlikely “one size fits all” paradigm
  - Importance of applications based upon “more than a single task”

- Extend traditional focus from end-users to workflow system/tool developers!
  - Building Blocks (BB) permit workflow tools and applications can be built.

- Need for agile, experimental and often unique workflows
  - Run many times, or many users: amortisation of development overhead
  - End-users develop interfaces, not performance critical components.
RADICAL’s Laws of CI (With apologies to Zawinski*)

- **RADICAL’s First Law**: Every tool “shims” to submit to distinct middleware (such as batch-queue systems) and claim interoperability.

*Sawinski’s Law: Every program attempts to expand until it can read mail. Those programs which cannot so expand are replaced by ones which can.
RADICAL’s Laws of CI

● RADICAL’s First Law: Every tool grows “shims” to submit to distinct middleware (such as batch-queue systems) and claim interoperability.

● Corollary: Interoperability should be provided explicitly at the lowest level possible (Principle of Subsidiarity)
RADICAL’s Laws of CI

- **RADICAL’s First Law**: Every tool grows “shims” to submit to distinct middleware (such as batch-queue systems) and claim **interoperability**.

- **Corollary**: Interoperability should be provided explicitly at the lowest level possible (Principle of Subsidiarity)

- **RADICAL’s Second Law**: Every application execution tool eventually claims to become a **workflow system**.
RADICAL’s Laws of CI

● **RADICAL’s First Law**: Every tool grows “shims” to submit to distinct middleware (such as batch-queue systems) and claim **interoperability**.

● **Corollary**: Interoperability should be provided explicitly at the lowest level possible (Principle of Subsidiarity)

● **RADICAL’s Second Law**: Every application execution tool eventually claims to become a **workflow system**.

● **Corollary**: To prevent proliferation of workflow systems we need to determine common components across (most) workflow systems.
RADICAL-Cybertools
RADICAL-Cybertools

- **Four Layers:**
  - L4: Application
  - L3: **Workload Management** (WLMS)
  - L2: **Task Run-time** (TRS)
  - L1: Resource Access Layer

- **Abstractions & Building Blocks:**
  - L1: **RADICAL-SAGA** Distributed job submission & standard interface
  - L2: **RADICAL-Pilot (RP)** Abstraction for Resource Management
  - L3: **RADICAL-WLMS, Ensemble Toolkit**

- **Cross-layer:** **RADICAL-Analytics**
RADICAL-Cybertools: “Building Blocks” for Workflows

- A “laboratory” while supporting production grade workflows and workflow tools.
- Stand alone, vertical integration and horizontal extensibility
- Integrated with existing tools:
  - Swift, Fireworks, PanDA, Binding Affinity Calculator (BAC)
  - Need “faster” start, “scalable” (more tasks) and “better” (resource utilization)
- Novel tools and libraries:
  - ExTASY, Replica-Exchange..
RADICAL-Cybertools: “Building Blocks” for Workflows

- A “laboratory” while supporting production grade workflows and workflow tools.
- Stand alone, vertical integration and horizontal extensibility
- Integrated with existing tools:
  - Swift, Fireworks, PanDA, Binding Affinity Calculator (BAC)
  - Need “faster” start, “scalable” (more tasks) and “better” (resource utilization)
- Novel tools and libraries:
  - ExTASY, Replica-Exchange..
RADICAL-Cybertools: “Building Blocks” for Workflows

- A “laboratory” while supporting production grade workflows and workflow tools.
- Stand alone, vertical integration and horizontal extensibility
- Integrated with existing tools:
  - Swift, Fireworks, PanDA, Binding Affinity Calculator (BAC)
  - Need “faster” start, “scalable” (more tasks) and “better” (resource utilization)
- Novel tools and libraries:
  - ExTASY, Replica-Exchange.

Fireworks + RP:
- Rockets start RP pilots on HPC hosts
- Rockets push tasks to RP for execution
RADICAL-Cybertools: “Building Blocks” for Workflows

- A “laboratory” while supporting production grade workflows and workflow tools.
- Stand alone, vertical integration and horizontal extensibility
- Integrated with existing tools:
  - Swift, Fireworks, PanDA, Binding Affinity Calculator (BAC)
  - Need “faster” start, “scalable” (more tasks) and “better” (resource utilization)
- Novel tools and libraries:
  - ExTASY, Replica-Exchange..
RADICAL-Cybertools: “Building Blocks” for Workflows

- A “laboratory” while supporting production grade workflows and workflow tools.
- Stand alone, vertical integration and horizontal extensibility
- Integrated with existing tools:
  - Swift, Fireworks, PanDA, Binding Affinity Calculator (HT-BAC)
  - Need “faster” start, “scalable” (more tasks) and “better” (resource utilization)
- Novel tools and libraries:
  - ExTASY, Replica-Exchange..
RADICAL-Cybertools: “Building Blocks” for Workflows

- A “laboratory” while supporting production grade workflows and workflow tools.
- Stand alone, vertical integration and horizontal extensibility
- Integrated with existing tools:
  - Swift, Fireworks, PanDA, Binding Affinity Calculator (HT-BAC)
  - Need “faster” start, “scalable” (more tasks) and “better” (resource utilization)
- Novel tools and libraries:
  - ExTASY, Replica-Exchange..
RADICAL Execution Model

- **AIMES Execution Model**: An *execution model* for dynamically federated heterogeneous resources that works *independent* of type of infrastructural dynamism and heterogeneity.

- **AIMES Model of Execution**:
  - Importance of *dynamic integration* of workload and resource information.
  - *Execution strategy*: Temporally ordered set of decisions that need to be made to execute a given workload.

- **RADICAL Execution Model**: Generalize AIMES Execution models to “better” and general mapping of workloads to infrastructure (in EnTK, ExTASY).
RADICAL-Pilot on Blue Waters
L2: Pilot-Abstraction (P* Model)

- ".. a scheduling overlay which generalizes the reoccurring concept of utilizing a placeholder as a container for compute tasks"
- Decouples workload from resource management
- Enables the fine-grained (ie “slicing and dicing”) of resources
- Tighter temporal control, advantages of application-level scheduling
- Build higher-level frameworks without explicit resource management
Agent: ORTE-LIB

- **ORTE: Open RunTime Environment**
  - Isolated layer used by Open MPI to coordinate task layout
  - Runs a set of daemons over compute nodes
  - No ALPS concurrency limits
- **Supports multiple tasks per node**
  - Uses library calls instead of `orterun` processes
  - No central fork/exec limits
  - Shared network socket
  - (Hardly) no central file system interactions
ExTASY Workflows on Blue Waters
The Power of Many: Ensemble Methods

- Many **sampling problems** formulated as ensemble methods/algorithms
- Ensemble members often interact.
  - Not a “bag-of-tasks” abstraction.
  - Replica-exchange, Adaptive Markov State Models, Enhanced Ensemble..
- Different degrees and levels of coupling between ensembles members
- Traditional HPC optimized for single large job(s).
Advanced Sampling Case Study: COCO

- **Better Sampling:** Drive systems towards unexplored regions, don’t waste time sampling behaviour already observed
  - E.g. DM-d-MD, COCO, …
- PCA-based Unsupervised Learning
Advanced Sampling: COCO

- **Better Sampling**: Drive systems towards unexplored regions, don’t waste time sampling behaviour already observed
  - E.g. DM-d-MD, COCO, ...
- PCA-based Unsupervised Learning
- DM-d-MD
COCO: PCA-based Unsupervised Learning
Sampling “Quality”

[Graphs showing the relationship between the number of CoCo-jumps in 2 ns and the volume sampled (ang^3) and mean median first passage time (ps).]

[Graph showing the relationship between the number of walkers and the volume sampled (ang^3) and mean median first passage time (ps).]
Performance

Amber simulations of the extasy workflow executed on NCSA Bluewaters

- **Time (seconds)**
  - 256,8192
  - 512,16384
  - 1024,32768
  - 2048,65536

- **No. of simulations, Cores (cores/sim = 32)**
In both cases, analysis is a single task using all the cores.

- Linear scaling observed
- Tested up to 4K cores
- Improvements in underlying framework will seamlessly propagate
Advocate a Building Blocks approach to Workflow Systems.
  ○ RADICAL-Cybertools are a realization of the Building Blocks approach to scalable workflows https://arxiv.org/abs/1609.03484

Demonstrate how abstractions and execution models unify HTC, D-HTC, HT-HPC and HPC under the same conceptual framework
  ○ Distinctions are by-products of specific cyberinfrastructure implementations; discussions around software misleading!
  ○ Principled design and development of general-purpose middleware.

ExTASY Workflows on Blue Waters demonstrate better sampling!