Middleware Building Blocks for Workflow Systems: The RADICAL-Cybertools Experience

Shantenu Jha, Andre Merzky and Matteo Turilli

http://radical.rutgers.edu

RADICAL-Cybertools: Motivation

- Initially workflow **management** systems provided “end-to-end” capabilities
  - Workflow systems were developed to support “big science” projects when software infrastructure was “fragile”, unreliable, missing services
- Workflows aren’t what they used to be!
  - High-throughput computing is important, but many other features
- **Upshot:** Need a sustainable ecosystem of **both** existing and new software components from which tailored workflow systems can be composed
  - Realize agile development and composition of workflow systems that leverage rich ecosystem of existing capabilities
  - **Understand needs of workflow system development, not just workflows!**

RADICALs' 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.**

*** Zawinski'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.**

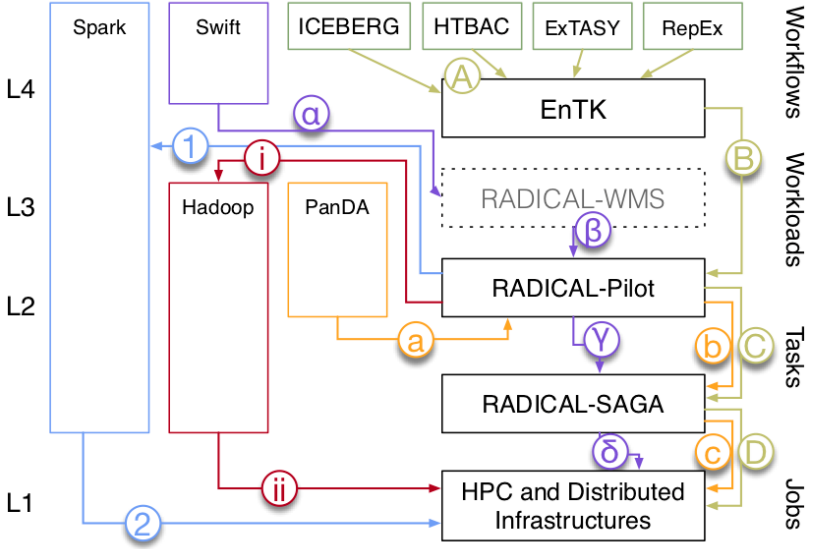
- **Corollary:** Interoperability should be provided explicitly at the lowest level possible (**Principle of Subsidiarity**)

- **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: Middleware Building Blocks

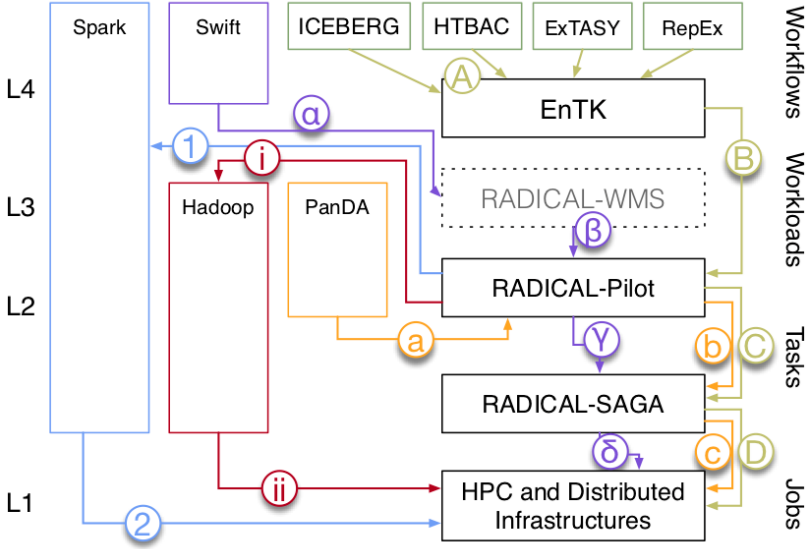
- A “laboratory” while supporting production grade workflows **and** workflow systems
- Stand alone, vertical and horizontal extensibility
- Integrate with existing tools:
  - PanDA, Swift, Fireworks, ...
  - Distinct points of integration
  - Need “faster” start, “scalable” (more tasks) and “better” (resource utilization)
- Novel tools and libraries:
  - ExTASY, RepEx, HTBAC, Seisflow,..



RADICAL-Cybertools as Building Blocks: Experience

Our preliminary experience with RADICAL-Cybertools (**RCT**) suggests:

- Building Blocks are easily extensible & can be used for proverbial last mile customization:
  - EnTK is used to support > 6 Domain Specific Workflows
  - RepEx, ExTASY, ICEBERG, HTBAC, SCALE-MS, INSPIRE, ExaLearn
- Building Blocks are compatible with HPC challenges and requirements
  - RP used on 66% of Titan; now Summit
- RCT integrate with many other components -- some HPC, others non-HPC (ABDS)
- ….



Middleware Building Blocks for Workflow Systems

Principled approach to the architectural design of middleware systems and applies traditional notion of modularity at the systems level to enable composability among independent software systems

- Software component agnostic to architectural, coordination, and communication patterns, implemented in an arbitrary programming language, and exposed via an API.
  - **Self-sufficient:** Implements set of functionalities; not depend on other blocks
  - **Composable:** Caller can compose functionalities from independent blocks.
  - **Interoperable:** Usable in diverse system without semantic modification
  - **Extensible:** Building block functionality and entities can be extended
- Work both individually, or integrated, or with third party software.

Middleware Building Blocks for Workflow Systems https://arxiv.org/abs/1903.10057

RCT Experience: Design is necessary, but not sufficient

- Argue that BB changes both the **technical** and **social factors** (incentives)
- **Technical: (i)** Understand needs of workflow system development; (**ii)** BB enables expert contributions, while lowering the breadth of expertise required of workflow system developers; (iii) performance is great differentiator
- **Social:** If ecosystem of components from which workflow systems are composed, less incentive to claim “my WMS” is `better`[ ] than “your WMS”!
- **Beyond BB:** (i) Work with infrastructure providers, scientific support teams; plant your students at computing centers :) (ii) Engagement and User driven co-design process; help them deliver science ASAP while formalizing formal requirements engineering processes; (iii) Build trust relationships and a community (e.g., ParSL fest); depending upon size, a governance structure beyond BDFL paradigm

Andre Merzky **Matteo Turilli**

Two leads who make it happen + Many RADICAL Lab Members

Join the team! Openings at all levels!