SAGA

A Simple API for Grid Applications

A Brief Introduction to SAGA

SAGA

A Simple API for Grid Applications

All material from this tutorial can be found at:

http://saga.cct.lsu.edu/software/cpp/documentation/tutorials/loni-training-2010

And at:

https://svn.cct.lsu.edu/repos/saga-projects/tutorial/general_tutorial

General Information and Documentation

- General information
  - http://saga. cct.lsu.edu/
- Documentation:
  - http://saga.cct.lsu.edu/software/cpp/documentation
- API documentation
  - Python
    - http://static.saga.cct.lsu.edu/apidoc/python/latest/
  - C++
    - http://static.saga.cct.lsu.edu/apidoc/cpp/latest/
- Programmers Guide:
  - https://svn.cct.lsu.edu/repos/saga/core/trunk/docs/manuals/programming_guide/tex/saga-programming-guide.pdf

SAGA

A Simple API for Grid Applications

Distributed Applications Development Challenges

- Developing Distributed Applications is fundamentally hard:
  - Intrinsic:
    - Control/Coordination & execution over Heterogeneous sites
    - Complex Design point/Models of Distributed Applications,
      - Reasons for using distributed CI -- more than (peak) performance result
  - Extrinsic:
    - (Complex) Underlying infrastructure & its provisioning
    - Large number Programming systems, tools and environments
      - Lack of well-defined interfaces & abstractions
      - Interoperability and extensibility become difficult
- Number of "effective" distributed applications that utilize resources sequentially, concurrently or asynchronously is low
  - Distributed CI: Is the whole > than the sum of the parts?
- See: DPA Survey Paper:
  - http://www.cct.lsu.edu/~sjha/dpa_publications/dpa_surveypaper.pdf

SAGA

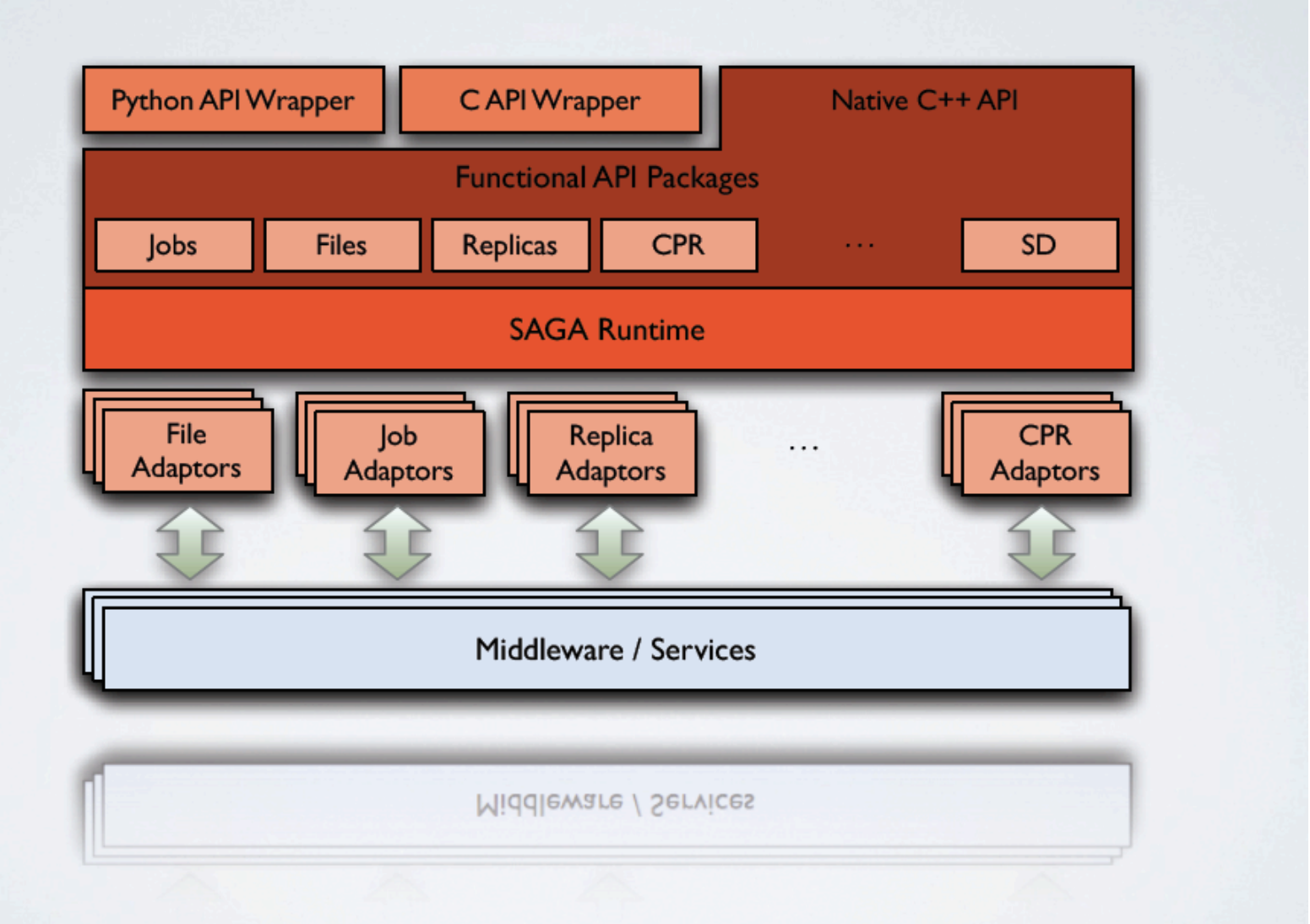
A Simple API for Grid Applications

SAGA: In a nutshell

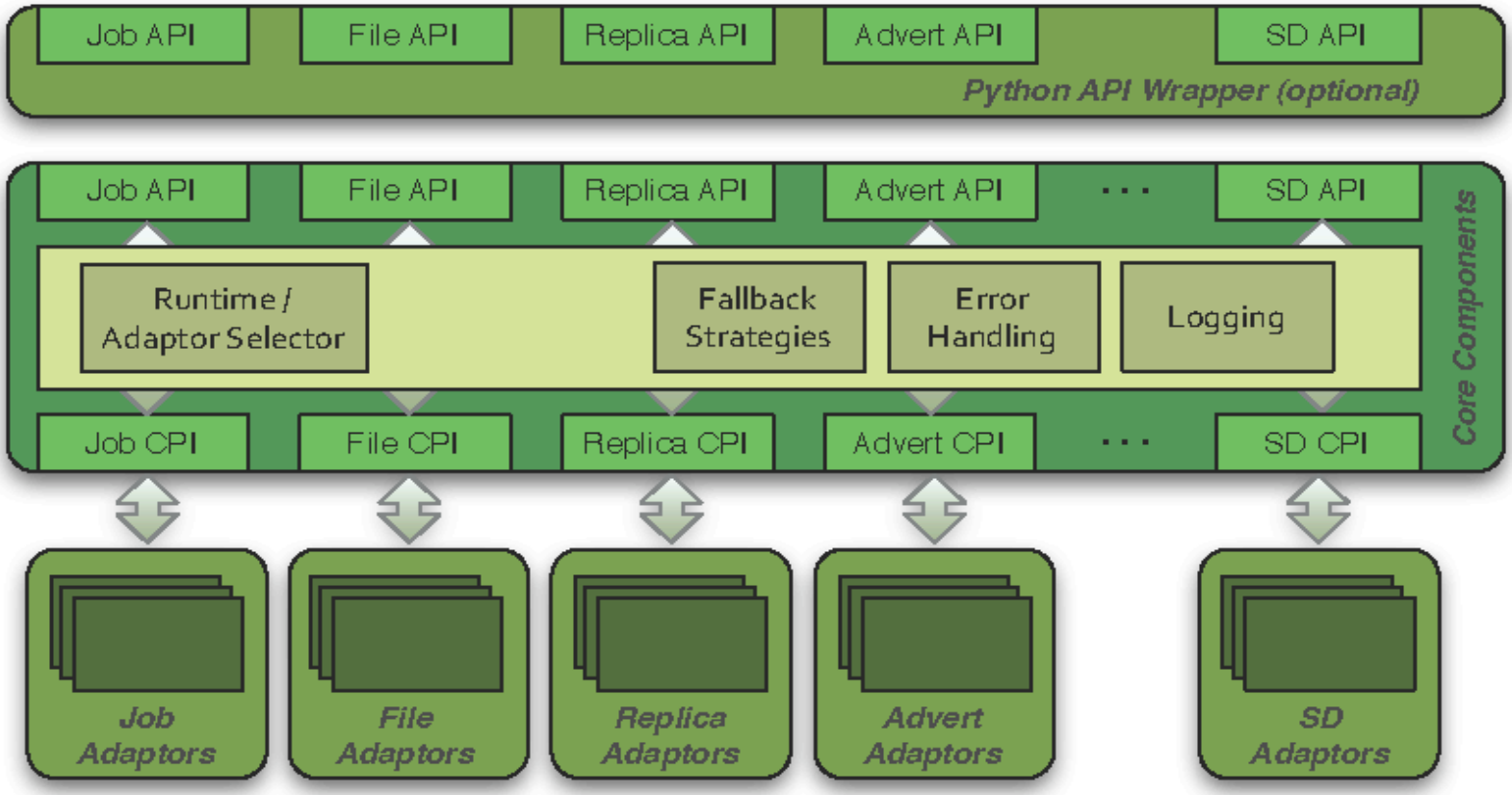
- There exists a lack of Programmatic approaches that:
  - Provide general-purpose, basic &common grid functionality for applications and thus hide underlying complexity, varying semantics..
  - The building blocks upon which to construct "consistent" higher-levels of functionality and abstractions
  - Meets the need for a Broad Spectrum of Application:
    - Simple scripts, Gateways, Smart Applications and Production Grade Tooling, Workflow…
- Simple, integrated, stable, uniform and high-level interface
  - Simple and Stable: 80:20 restricted scope and **Standard**
  - Integrated: Similar semantics & style across
  - Uniform: Same interface for different distributed systems
- SAGA: Provides Application* developers with units required to compose high-level functionality across (distinct) distributed systems

  (*) One Person's Application is another Person's Tool

SAGA: In a thousand words



SAGA: Architecture



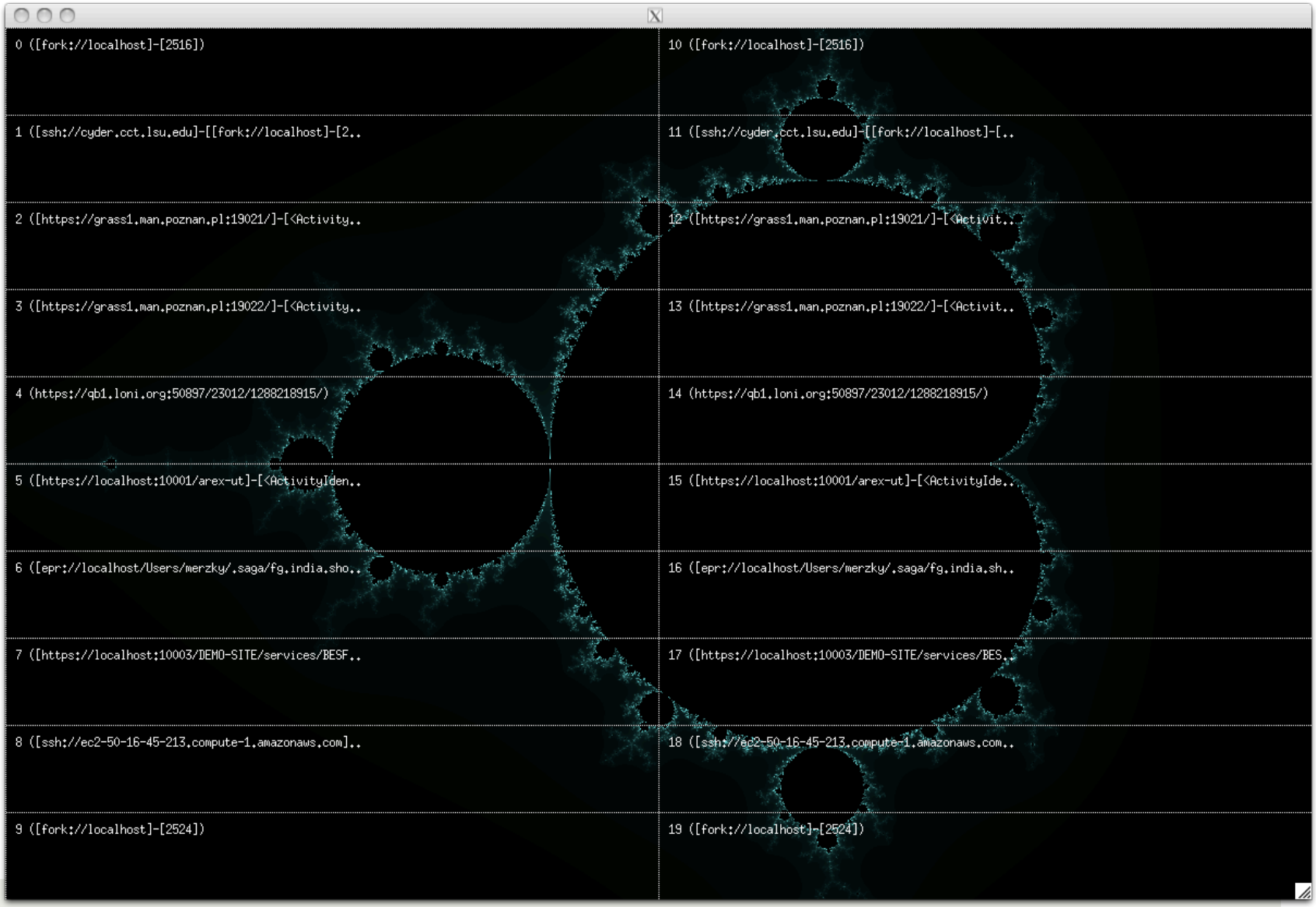
SAGA
A Simple API for Grid Applications

How is SAGA Used?

- SAGA is used to develop applications that are distributed by definition:
  - Simple extensions of "localized applications" (eg scripting)
    - MW applications, workers submitted to >8 back-ends
  - Novel Distributed Programming Models (eg Rep-Exch)
- SAGA: Build tools and implement abstractions that enable the execution of applications over distributed resources, *without modifying the applications*
  - Eg. Infrastructure Independent Pilot-Jobs
- SAGA: To provide uniform access layers to heterogeneous CI
  - Uniform access to EGI (ARC, gLite, Globus and Unicore/BES)
  - Simplify the building of tools and Gateways

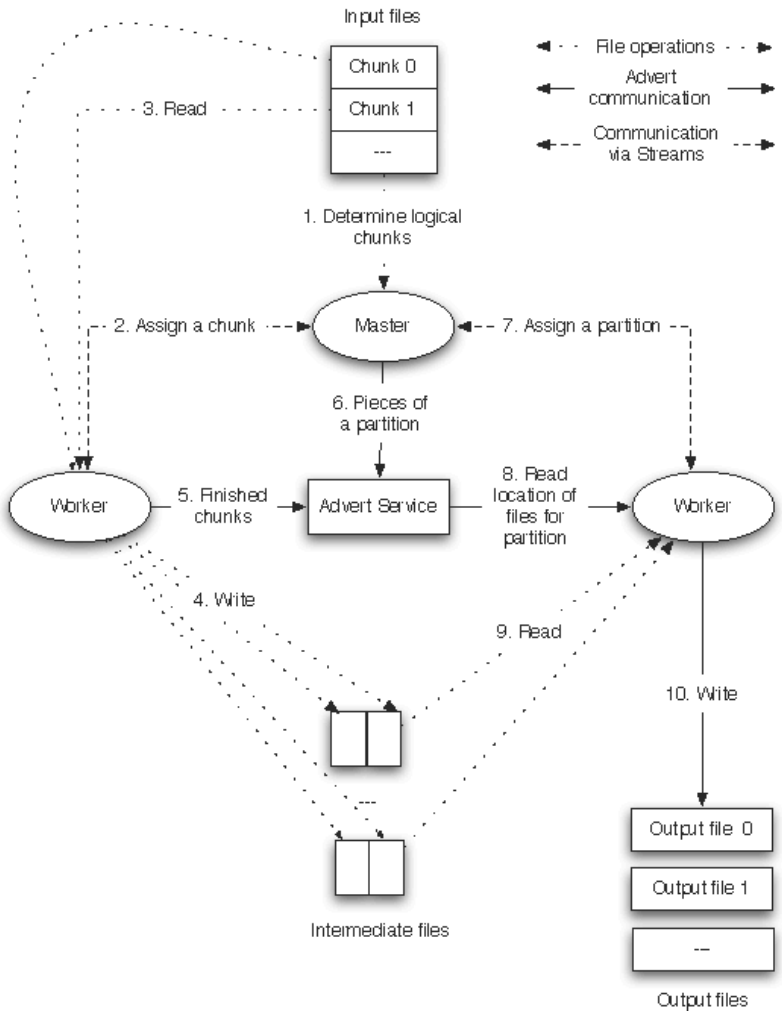
CENTER FOR COMPUTATION & TECHNOLOGY

1. Develop applications that are distributed by definition



SAGA: Develop applications that are distributed by definition

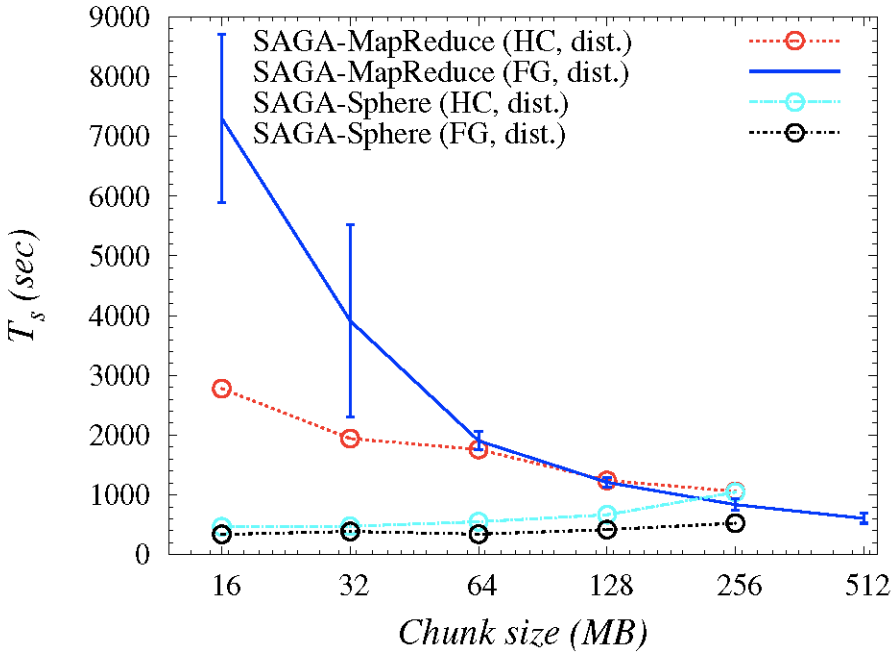
- How to develop a simple MR that is interoperable across infrastructure concurrently?
- Same application, same programming model:
  - Very different performance dependence
- Same application, different programming models
  - Very different performance dependence

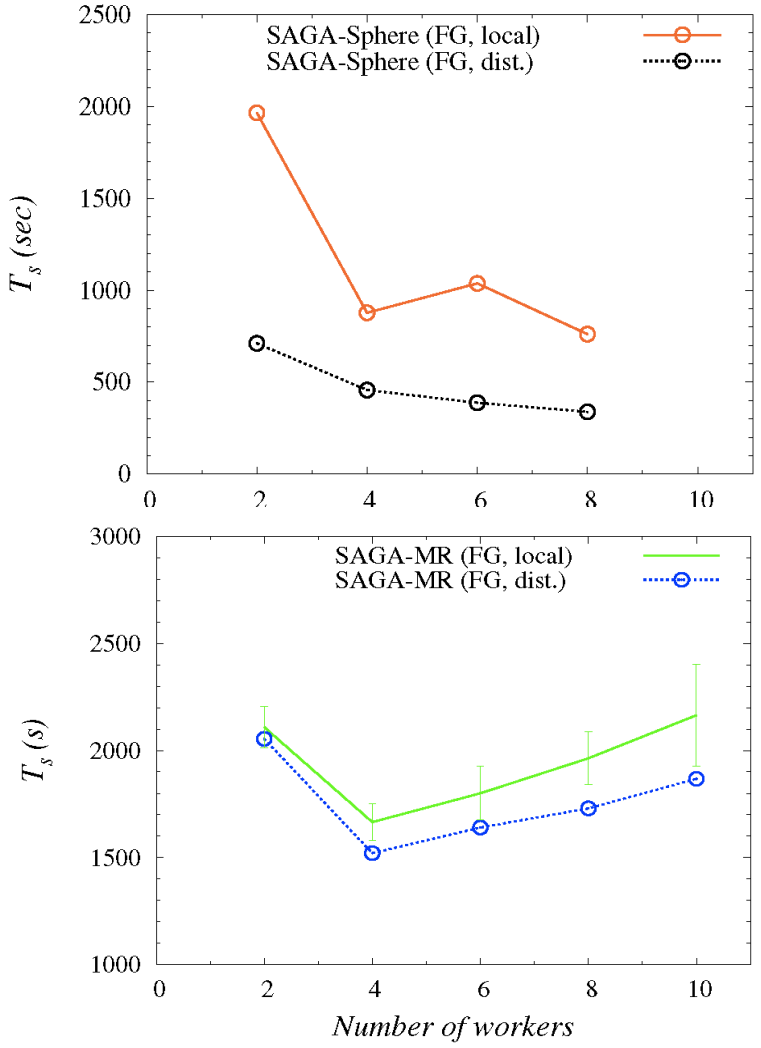


SAGA

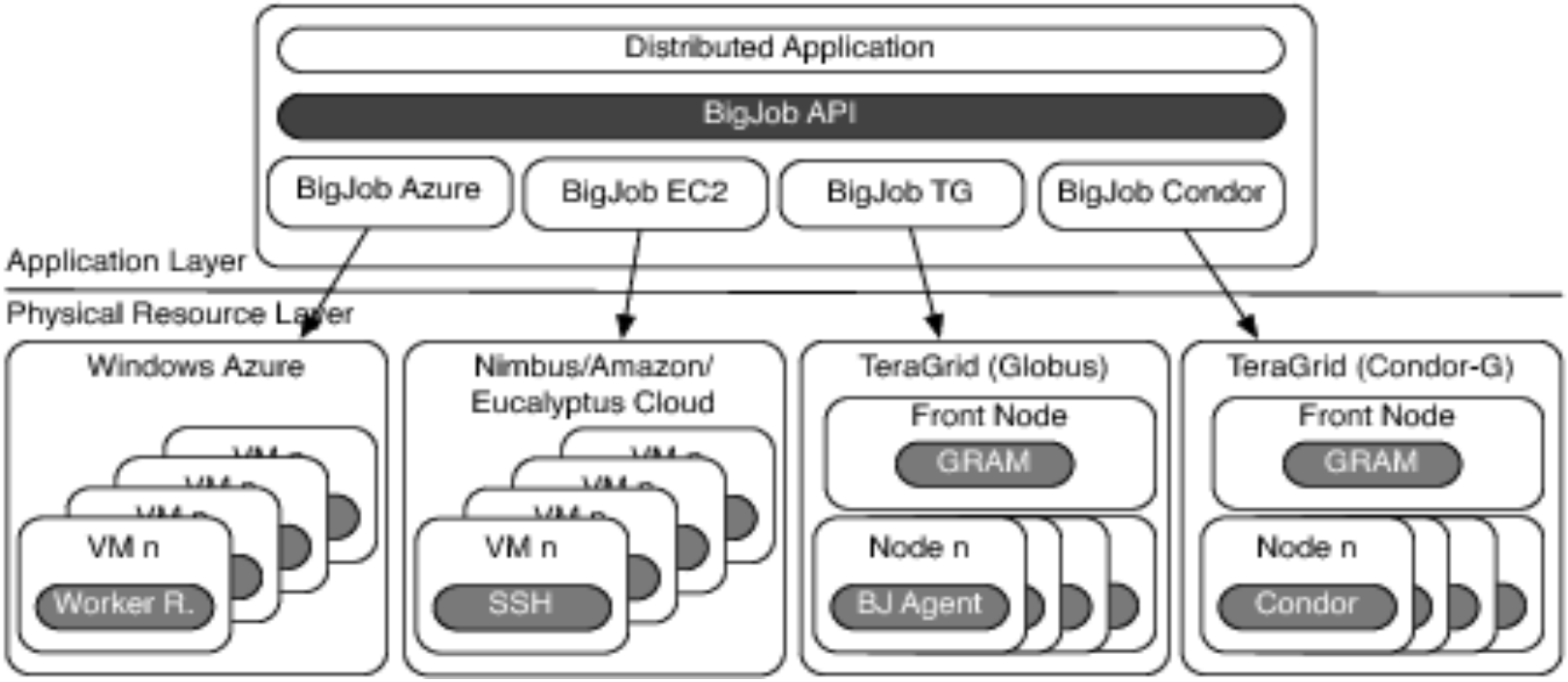
A Simple API for Grid Applications

Understanding Distributed Programming Models





2. Tools for Effective Distributed Execution

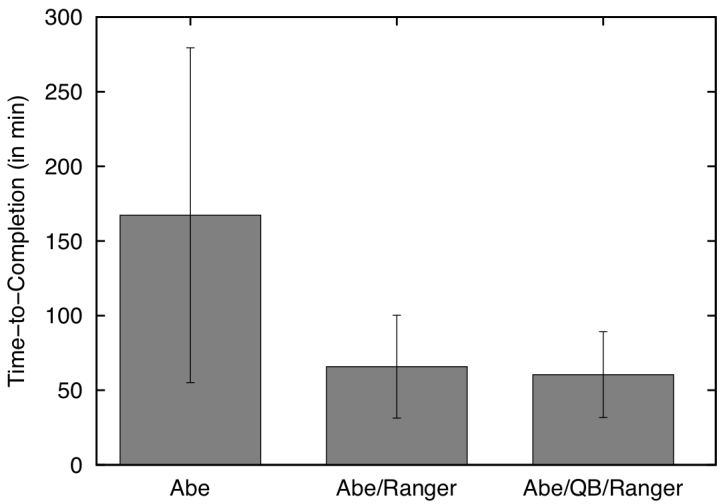


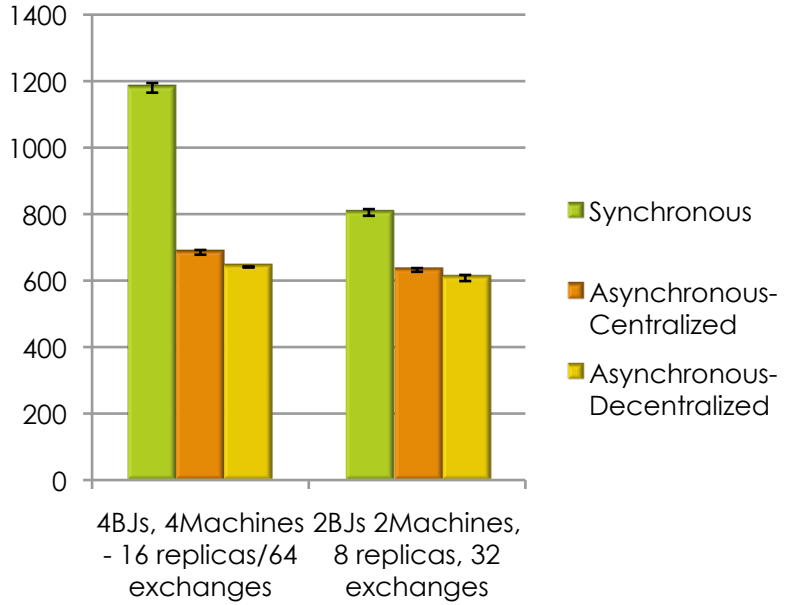
SAGA

A Simple API for Grid Applications

Distributed Adaptive Replica Exchange (DARE) Multiple Pilot-Jobs on the "Distributed" TeraGrid

- Ability to dynamically add HPC resources. On TG:
- Innovations in Distributed Algorithms:
  - Variants of RE: Sync (local) vs async (distr.)





3. Provides uniform access layers to heterogeneous CI

NGS
National Grid Service

Distributed
European
Infrastructure for
Supercomputing
Applications

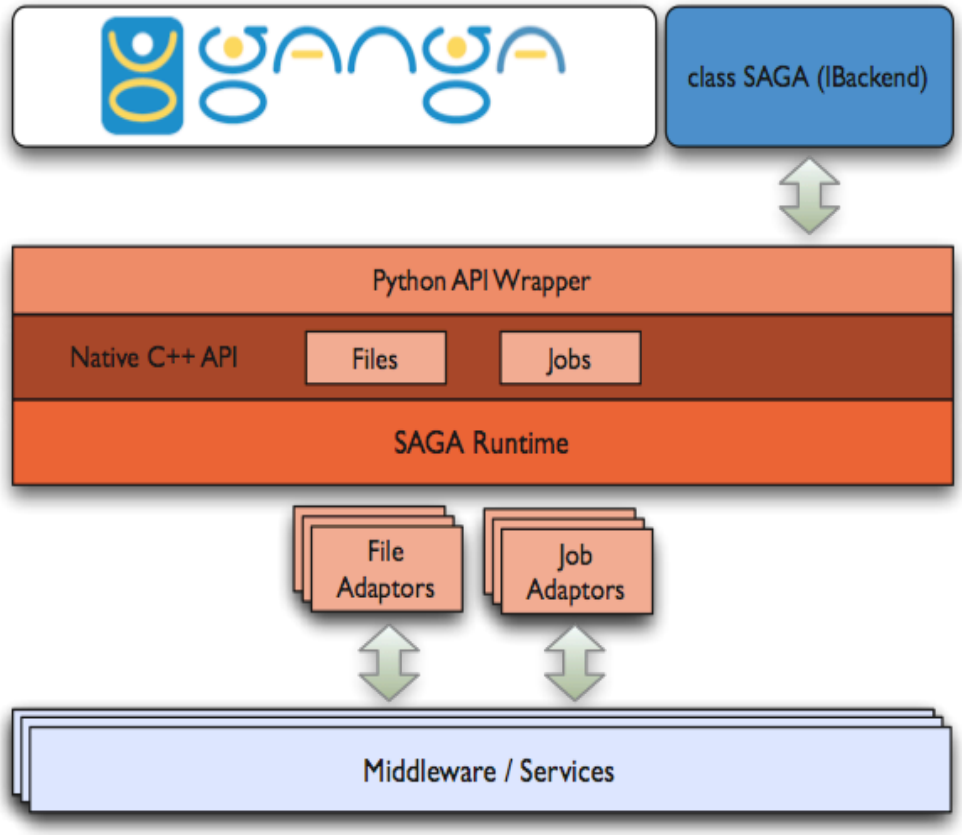
TeraGrid™
EMPOWERING DISCOVERY

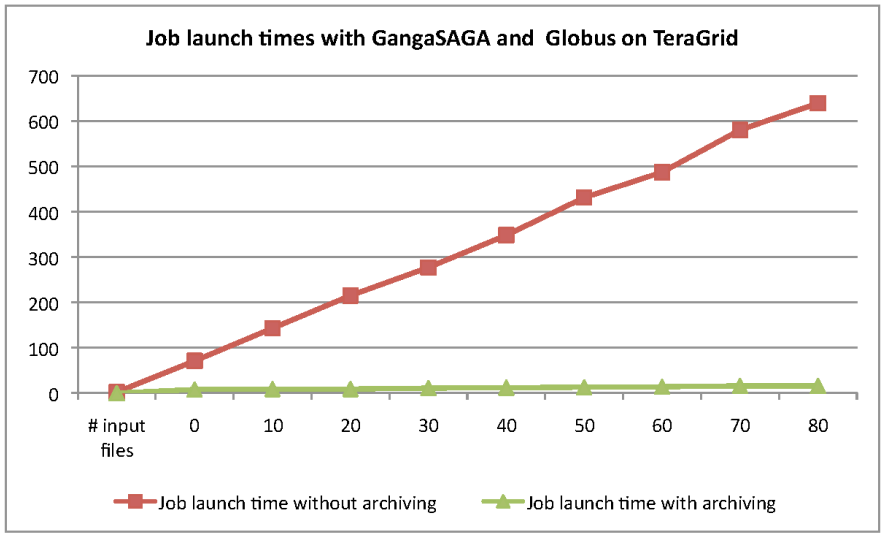
Grid Applications
Grid Middleware
Networking
NAREGI

eGee
Enabling Grids
for E-sciencE

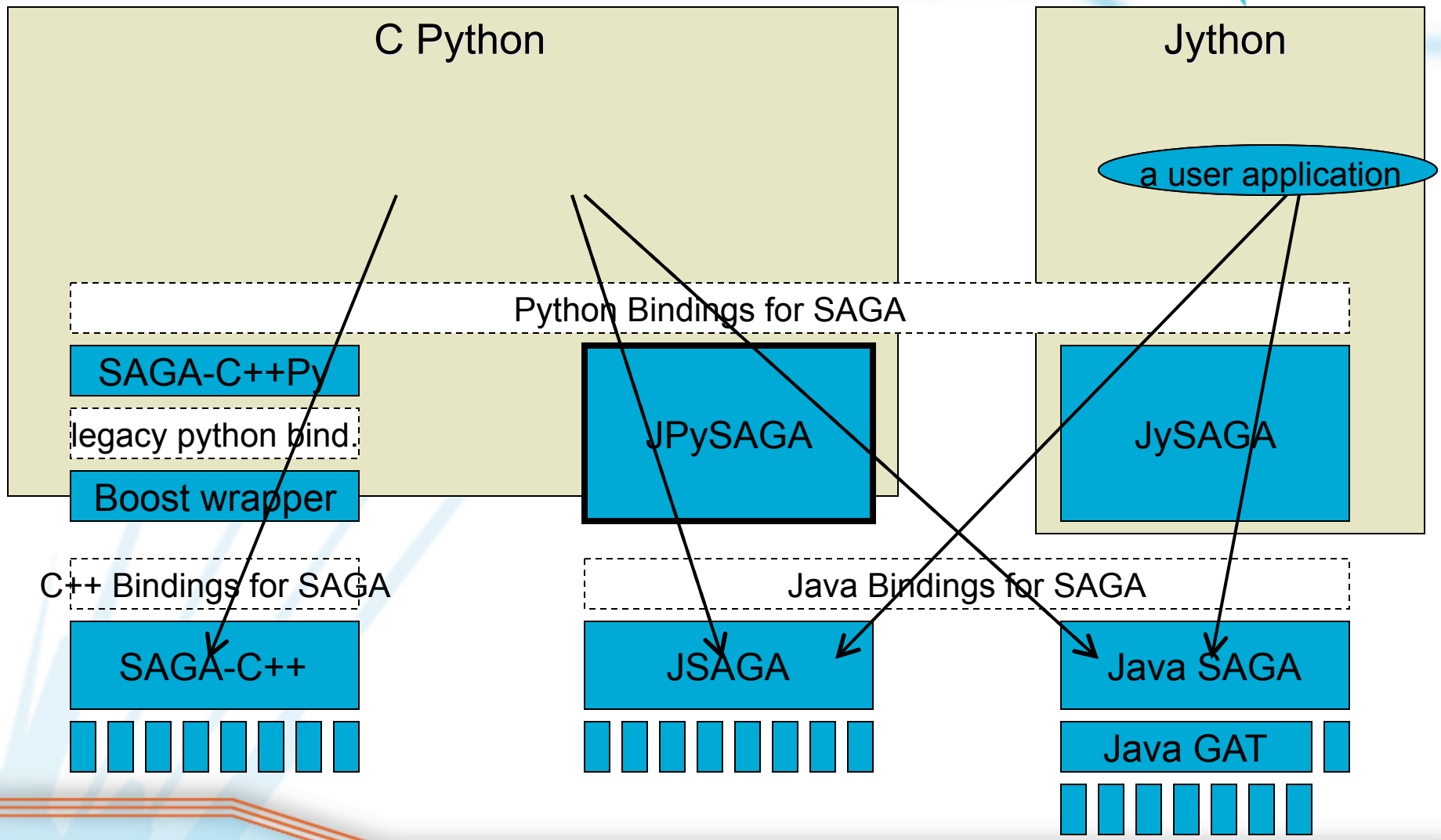
CENTER FOR COMPUTATION & TECHNOLOGY

SAGA-GANGA Integration





Java-based Python SAGA wrapper



CENTER FOR COMPUTATION & TECHNOLOGY

DARE – Gateway for RNA-folding
(Joohyun Kim, CyD)

- DARE-Gateway:
  - Integrated, Extensible
  - Balanced: Scale-Up and Scale-out to
- DARE-RFOLD, DOCK, Bioscope (NG Sequence Data), STMD (Molecular Dynamics)