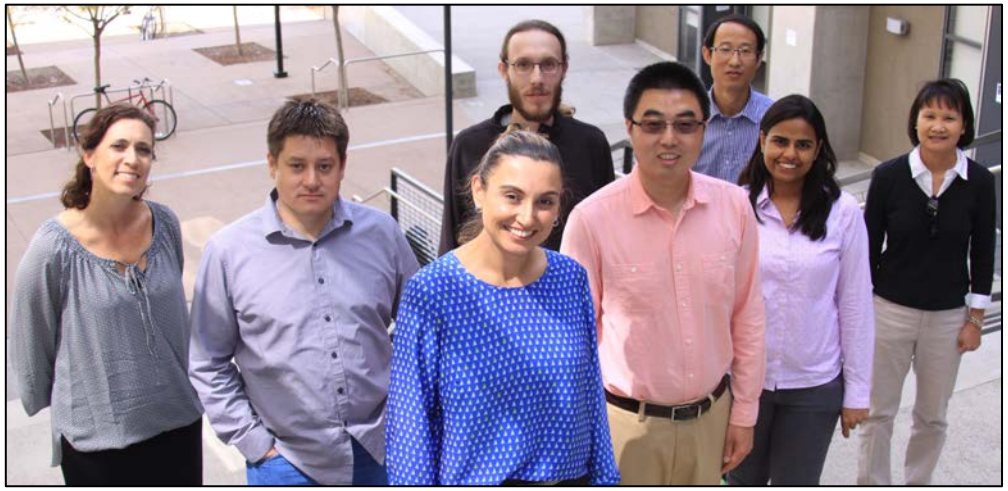
A Distributed Data-Parallel Execution Framework in the Kepler Scientific Workflow System

Ilkay Altintas and Daniel Crawl
San Diego Supercomputer Center
UC San Diego

Jianwu Wang
UMBC

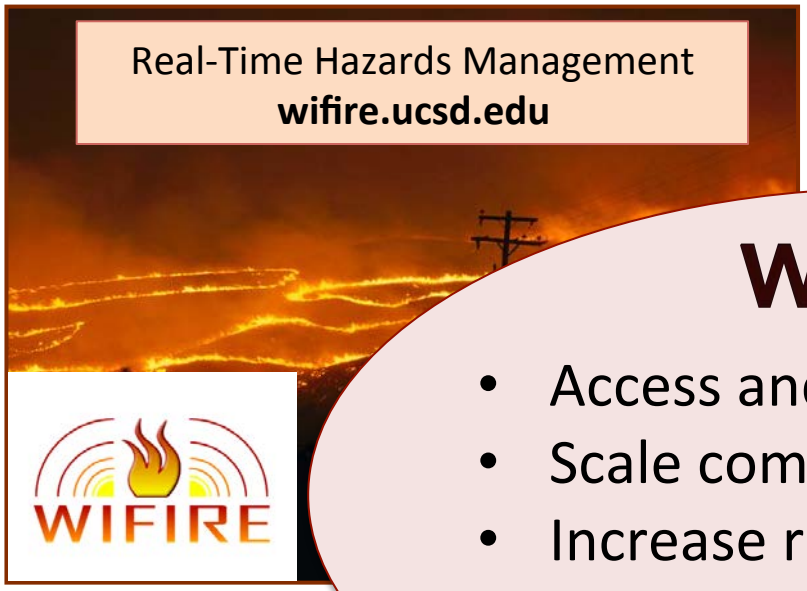


SDSC
SAN DIEGO SUPERCOMPUTER CENTER

WorDS.sdsc.edu

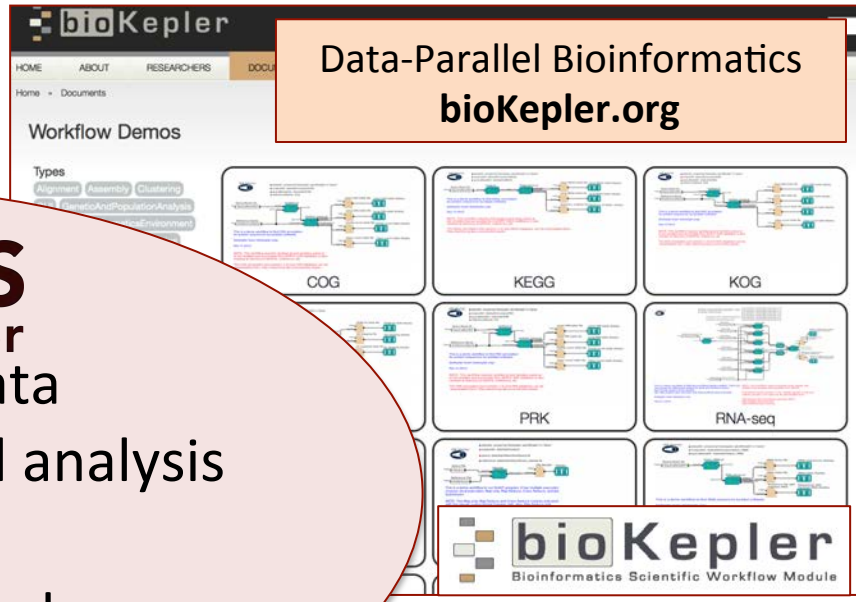
WorDS Center

Computational and Data Science Workflows
- **Programmable** and **Reproducible Scalability** -



Real-Time Hazards Management
wifire.ucsd.edu

WIFIRE

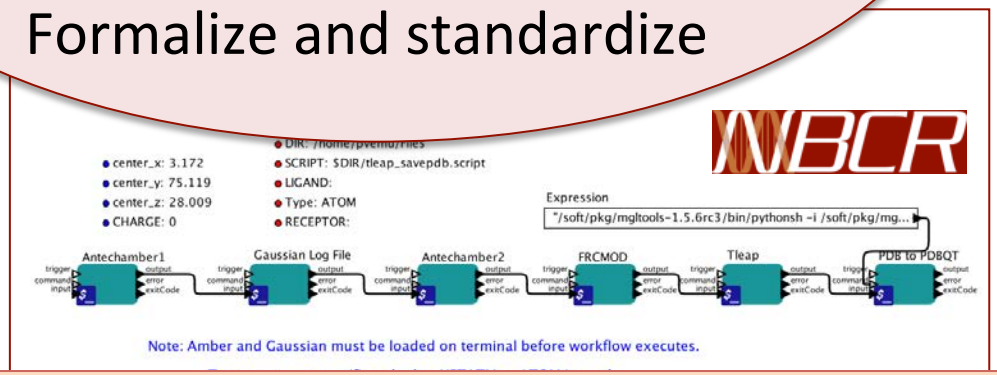


Data-Parallel Bioinformatics
bioKepler.org

bioKepler
Bioinformatics Scientific Workflow Module

WorDS Center

- Access and query data
- Scale computational analysis
- Increase reuse
- Save time, energy and money
- Formalize and standardize



NBCR

kepler-project.org

Kepler

Scalable Automated Molecular Dynamics and Drug Discovery
nbcr.ucsd.edu

WorDS.sdsc.edu

WorDS Center

Kepler is a Scientific Workflow System

www.kepler-project.org

- A cross-project collaboration
  … initiated August 2003
- 2.5 about to be released

- Builds upon the open-source Ptolemy II framework

Ptolemy II: A laboratory for investigating design

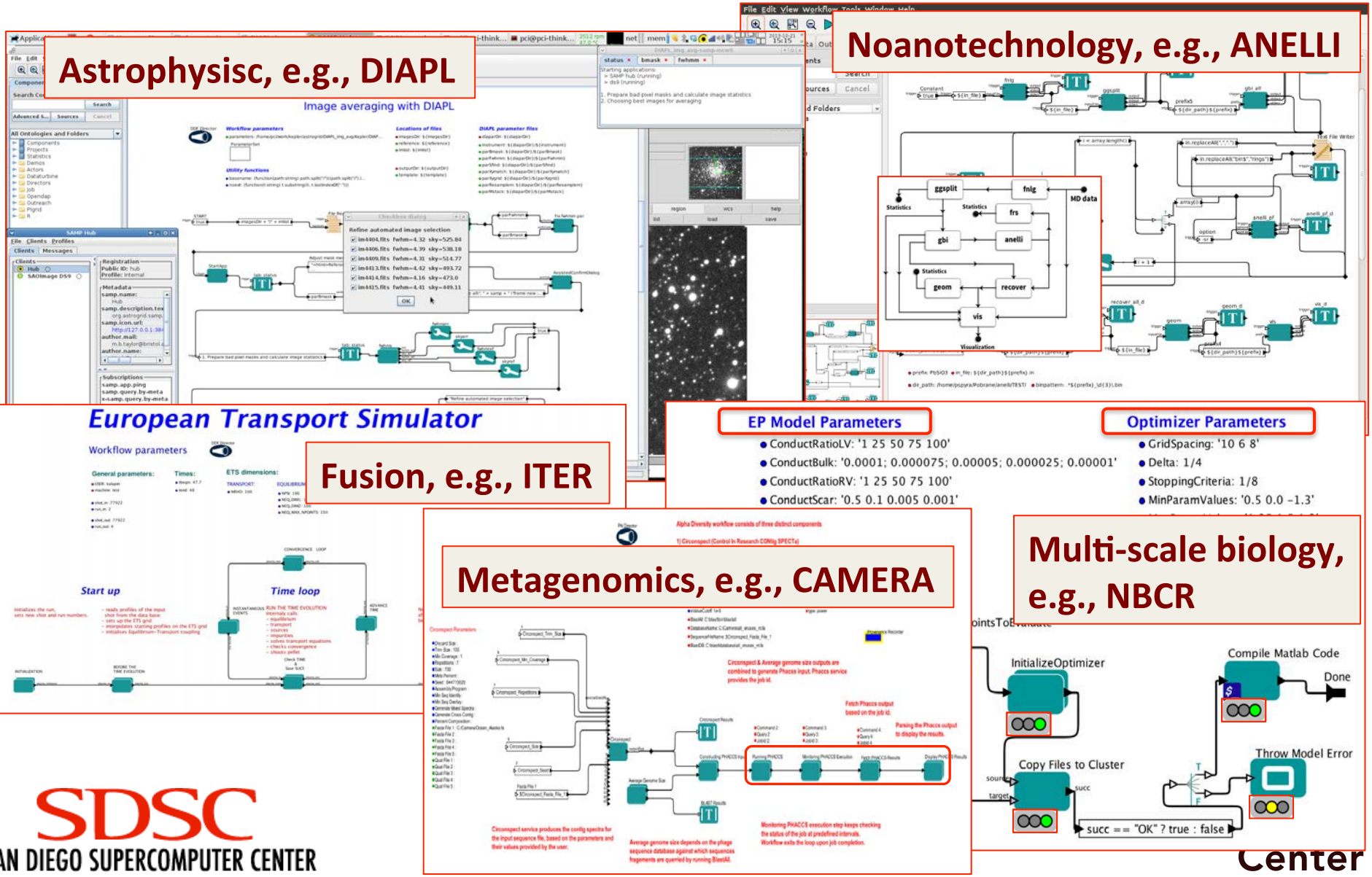
KEPLER: A problem-solving environment for workflow management

KEPLER = "Ptolemy II + X" for Scientific Workflows

SDSC
SAN DIEGO SUPERCOMPUTER CENTER

WorDS Center

Kepler was applied to problems in different scientific disciplines: some here and many more…



SDSC
SAN DIEGO SUPERCOMPUTER CENTER

Center

So,
how can we use Kepler workflows in the context of big data applications?

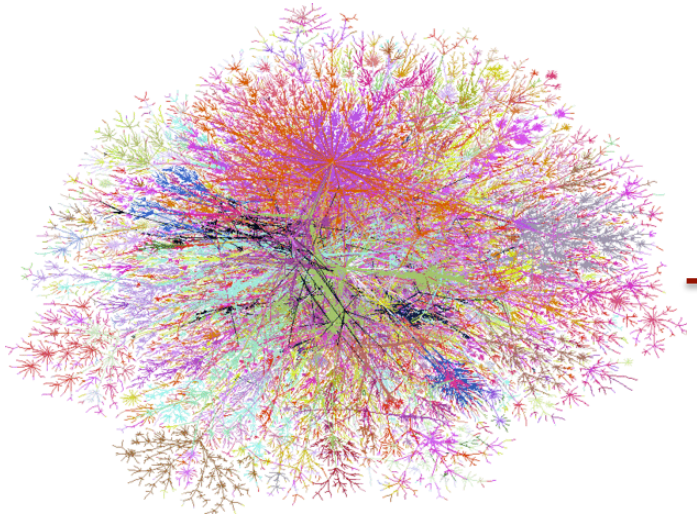
… while coupling all scales computing computing within a reusable solution…

SDSC
SAN DIEGO SUPERCOMPUTER CENTER

WorDS Center

APPLICATION-SPECIFIC KNOWLEDGE and QUESTIONS

?



BIG DATA

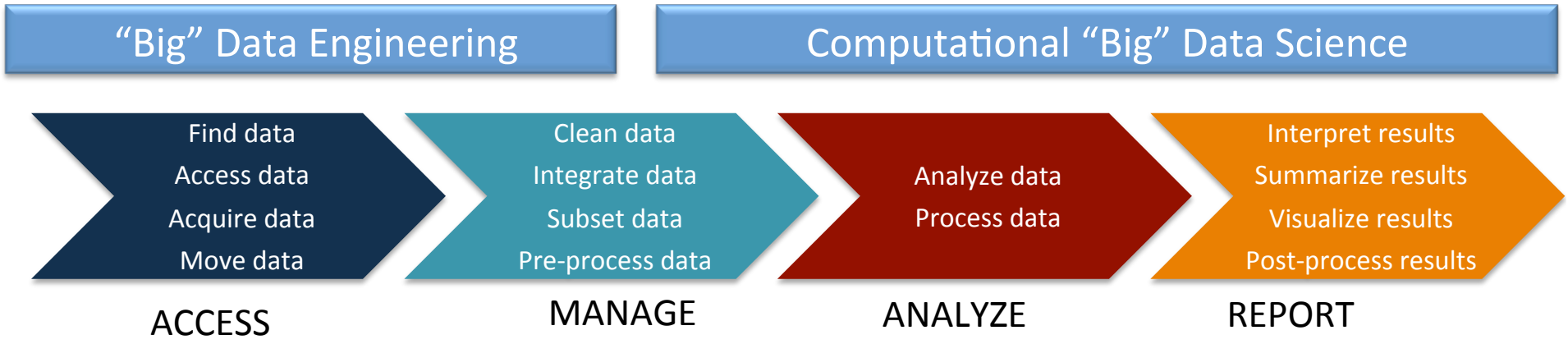
ON-DEMAND COMPUTE

Allows for data-enabled decision making at scale, using statistics, data mining, graph analytics, but also computational science methods.

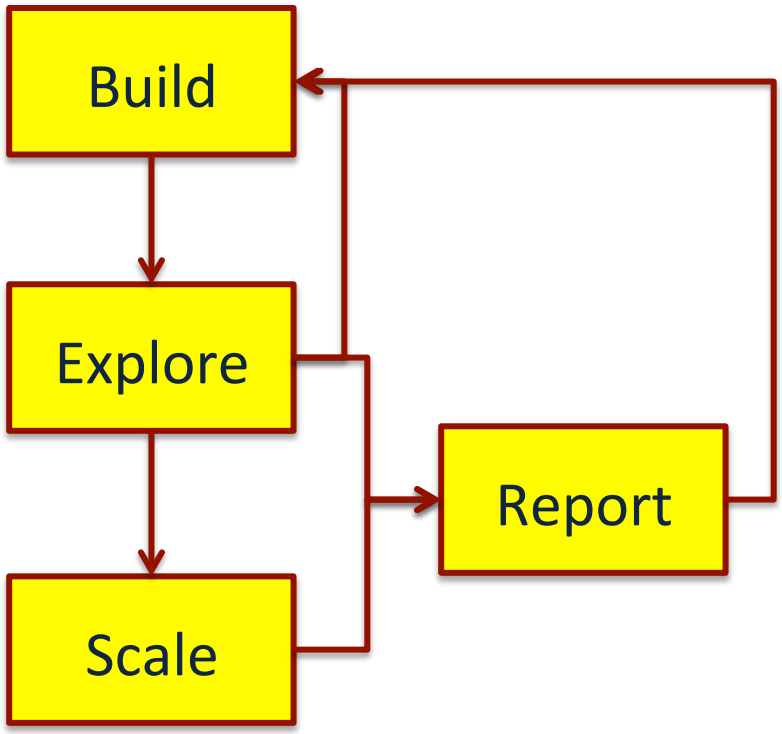
Requires support for experimental work by a multidisciplinary group of experts and dynamic scalability on many platforms!

SDSC
SAN DIEGO SUPERCOMPUTER CENTER

WorDS Center



Many ways to look at the process… not every step is automatable!



SDSC
SAN DIEGO SUPERCOMPUTER CENTER

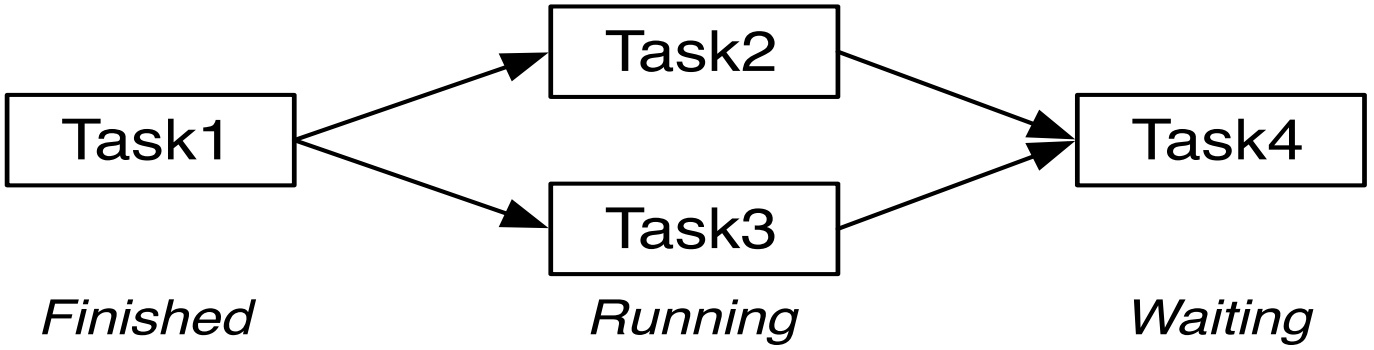
WorDS Center

Build Once, Run Many Times…

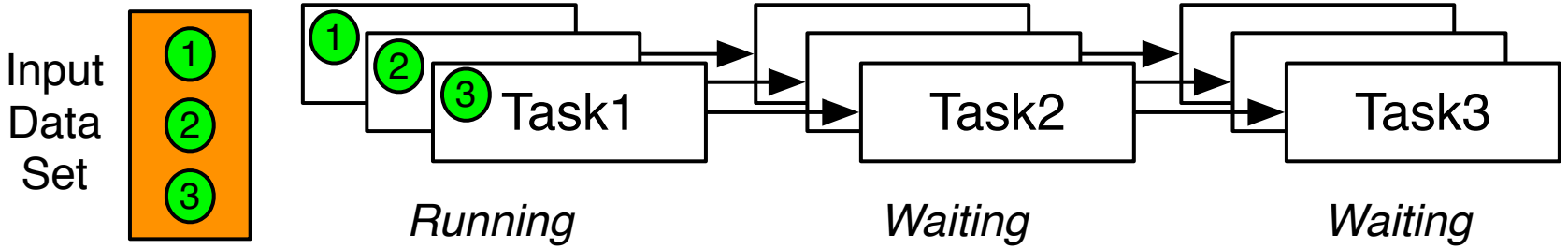
- Big data workflows should support experimental work and dynamic scalability on many platforms

- Scalability based on:
  - data volume and velocity
  - dynamic modeling needs
  - highly-optimized HPC codes
  - changes in network, storage and computing availability

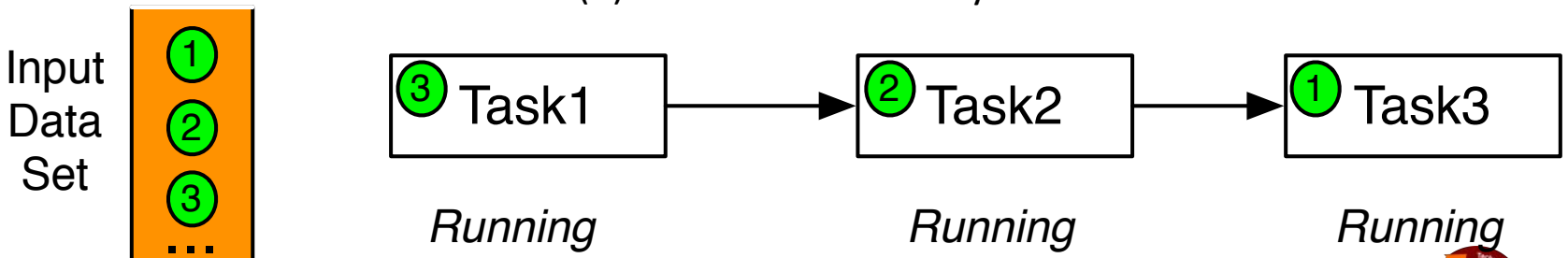
Scalability in Workflow



(a) Task-level scalability

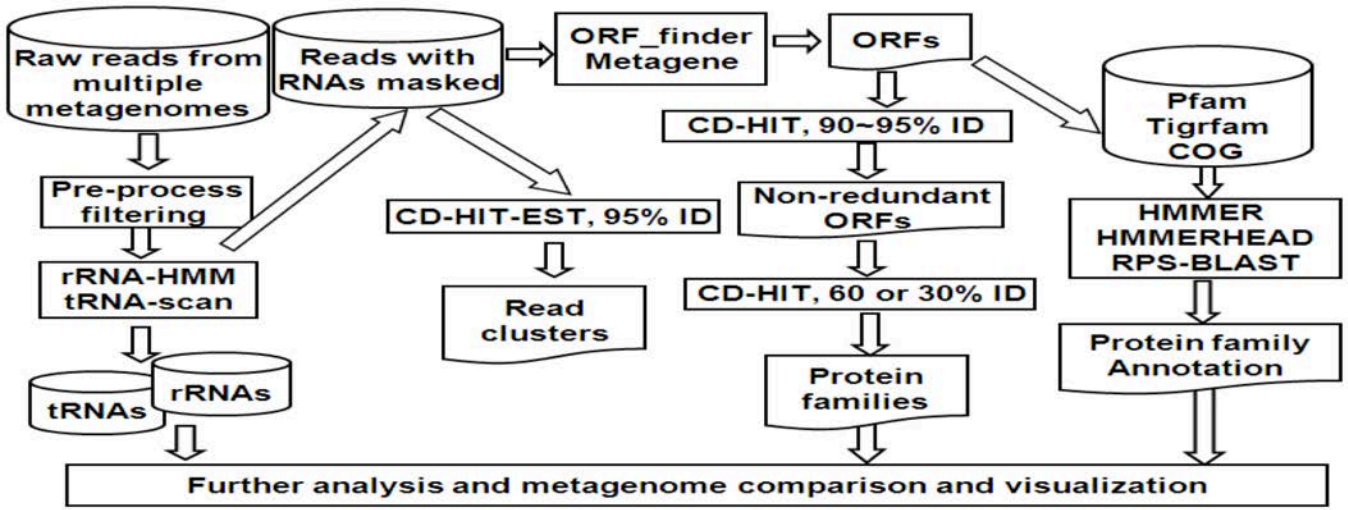


(b) Data-level scalability

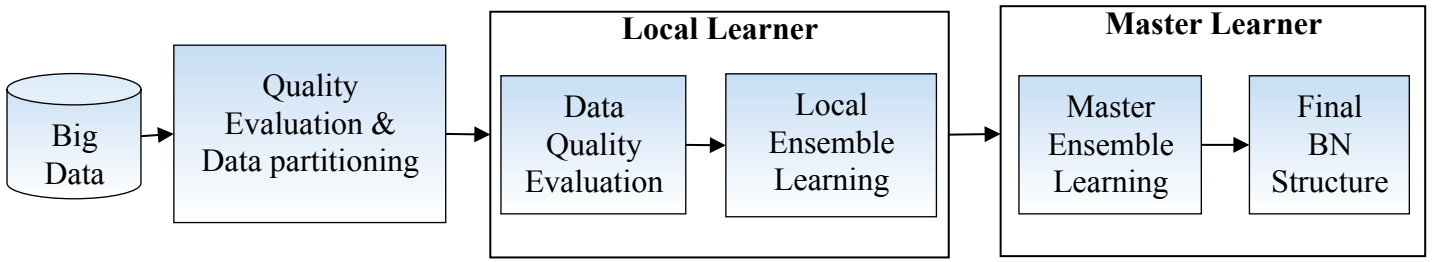


(c) Pipeline-level scalability

Big Batch Data Application Examples



RAMMCAP: Rapid Analysis of Multiple Metagenomes with Clustering and Annotation



BNL: Big Data Bayesian Network Ensemble Learning

SDSC
SAN DIEGO SUPERCOMPUTER CENTER

WorDS Center

Characteristics of the Big Data Applications

Name	Data Size	Tasks	Computational Requirements
RAMMCAP	7~14 million reads and much more with next generation sequencing (NGS) techniques 10~100 GB	12 bioinformatics tools	Some are computation intensive; some are memory intensive. All run on one node
BNLA	5 ~ 100 million records 10~1000 GB	3 programs written in R	Run on one node

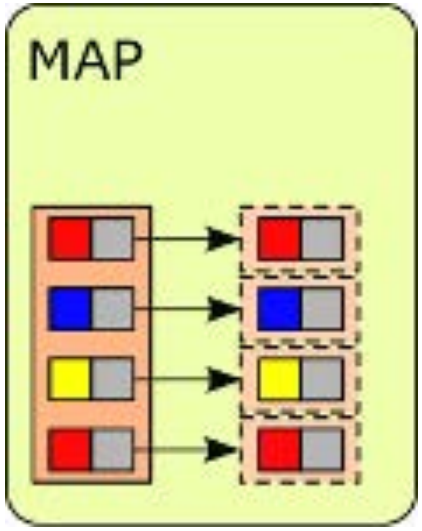
Goal: How to easily build efficient Big Data applications?

SDSC
SAN DIEGO SUPERCOMPUTER CENTER

WorDS Center

Distributed-Data Parallel Computing

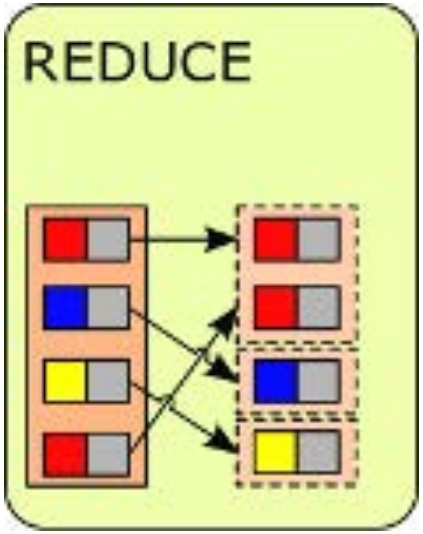
- A parallel and scalable programming model for Big Data
  - Input data is automatically partitioned onto multiple nodes
  - Programs are distributed and executed in parallel on the partitioned data blocks



MapReduce

Move program to data!

Images from:
http://www.stratosphere.eu/projects/Stratosphere/wiki/PactPM



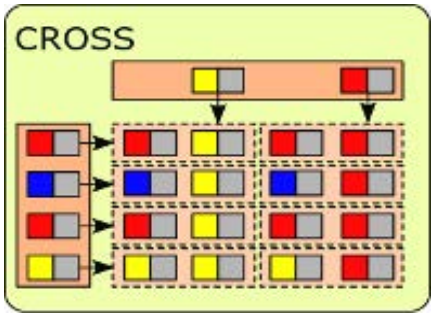
SDSC
SAN DIEGO SUPERCOMPUTER CENTER

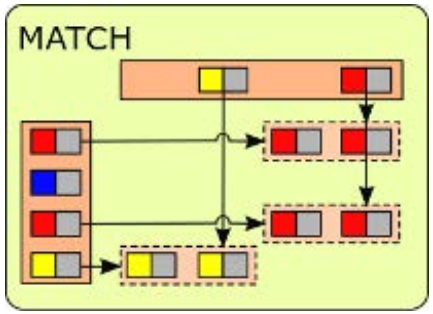
WorDS Center

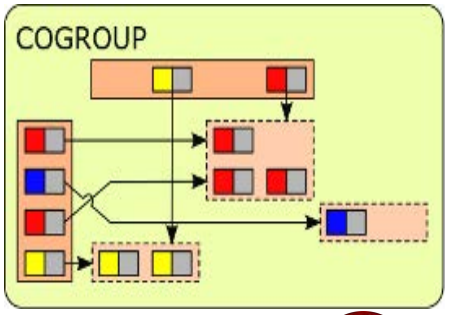
Distributed Data-Parallel (DDP) Patterns

Patterns for **data distribution** and **parallel data processing**

- A higher-level programming model
  - Moving computation to data
  - Good scalability and performance acceleration
  - Run-time features such as fault-tolerance
  - Easier parallel programming than MPI and OpenMP

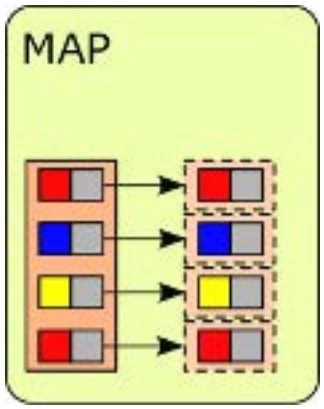


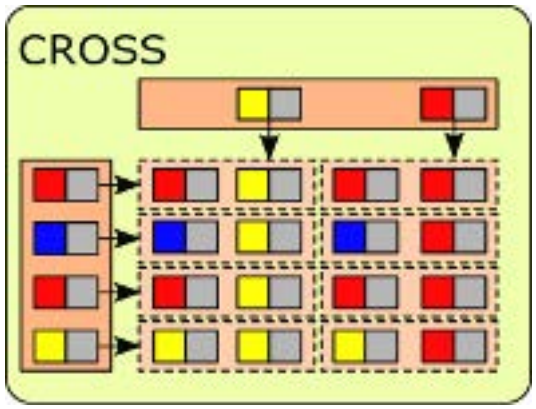


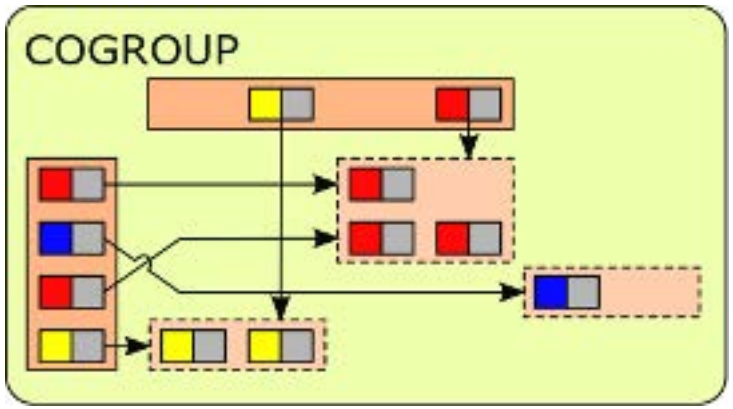


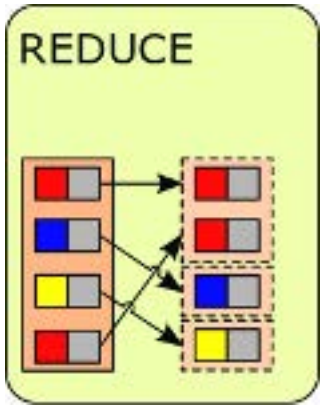
Images from: http://www.stratosphere.eu/projects/Stratosphere/wiki/PactPM

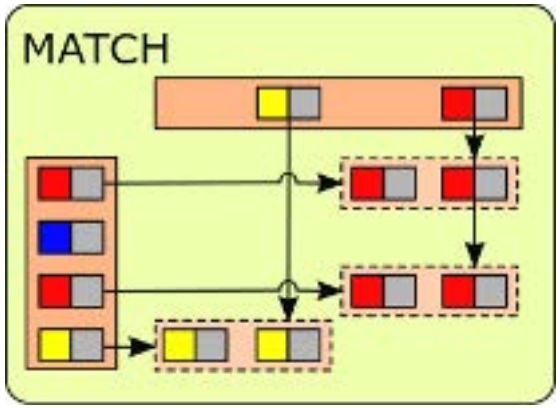
Some Available Big Data Patterns (a.k.a., Distributed Data-Parallel Patterns)







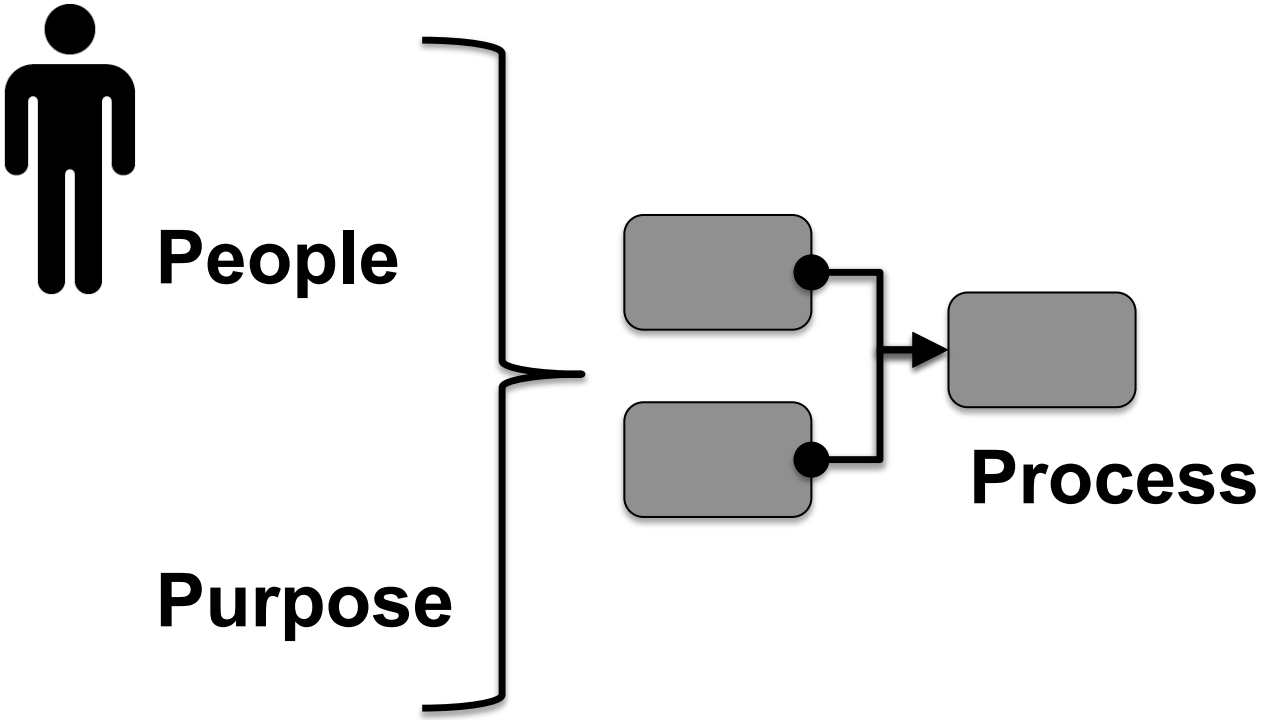




- The executions of the patterns can scale in distributed environments
- The patterns can be applied to different user-defined functions

Images from: http://www.stratosphere.eu

Scalability across platforms…



Platforms

SDSC
SAN DIEGO SUPERCOMPUTER CENTER

WorDS Center

Hadoop

- Open source implementation of MapReduce
- A distributed file system across compute nodes (HDFS)
  - *Automatic data partition*
  - *Automatic data replication*
- Master and workers/slaves architecture
- Automatic task re-execution for failed tasks

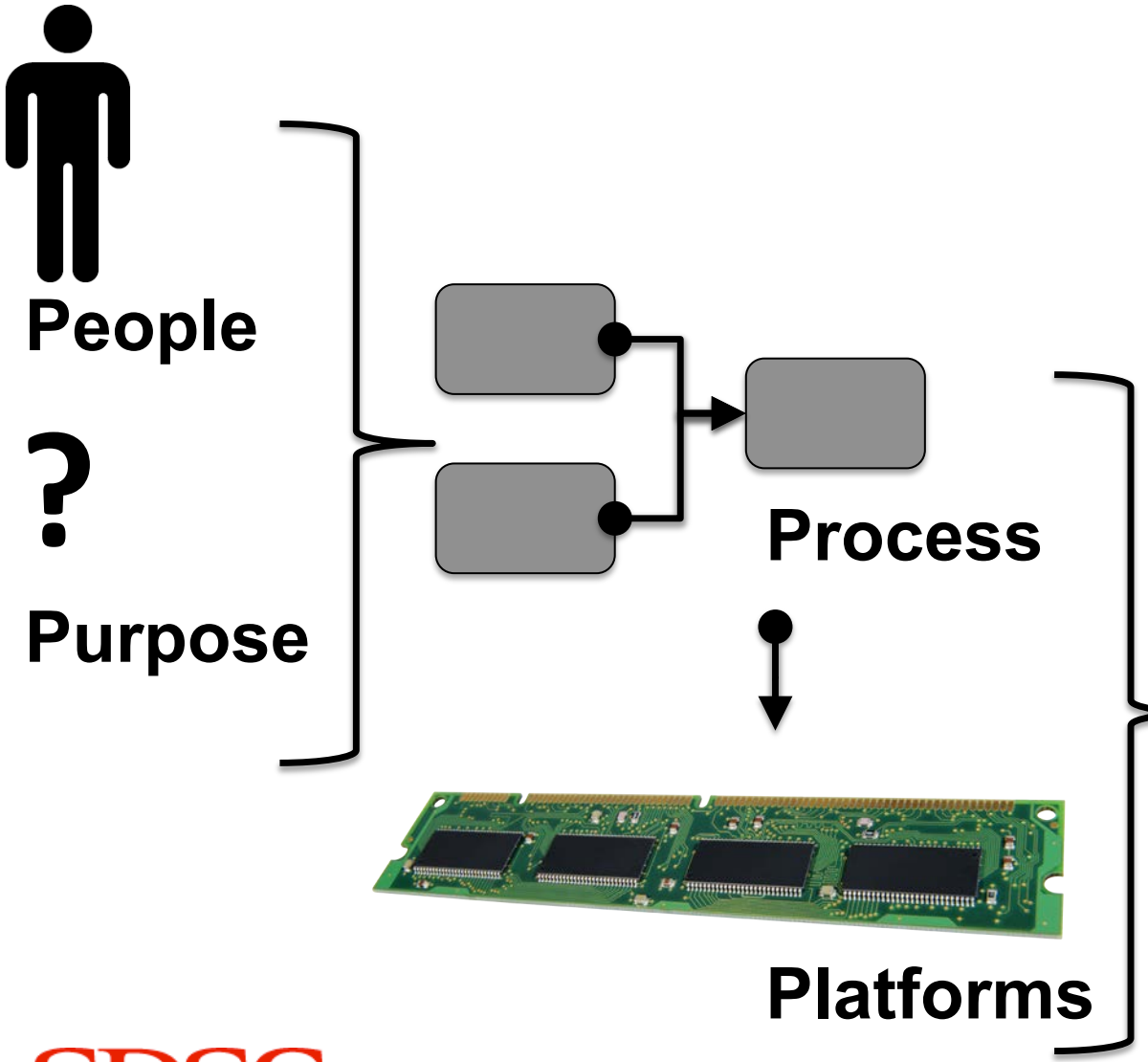
Spark

- Fast Big Data Engine
  - Keeps data in memory as much as possible
- Resilient Distributed Datasets (RDDs)
  - Evaluated lazily
  - Keeps track of lineage for fault tolerance
- More operators than just Map and Reduce
- Can run on YARN (Hadoop v2)

SDSC
SAN DIEGO SUPERCOMPUTER CENTER

WOrDS Center

The scalable Process should be Programmable!



Programmability

Reusable and reproducible programming interfaces to systems middleware, analytical tools, and end user&reporting environments.

SDSC
SAN DIEGO SUPERCOMPUTER CENTER

WorDS Center

Challenge Analysis

- How to easily apply the Big Data Patterns in a workflow?
- How to parallelize legacy tools for Big Data?
- How to pick pattern(s) each specific task/tool?
- How to run the same process on top of different Big Data engines, such as Hadoop, Spark and Stratosphere (Apache name: Flink)?

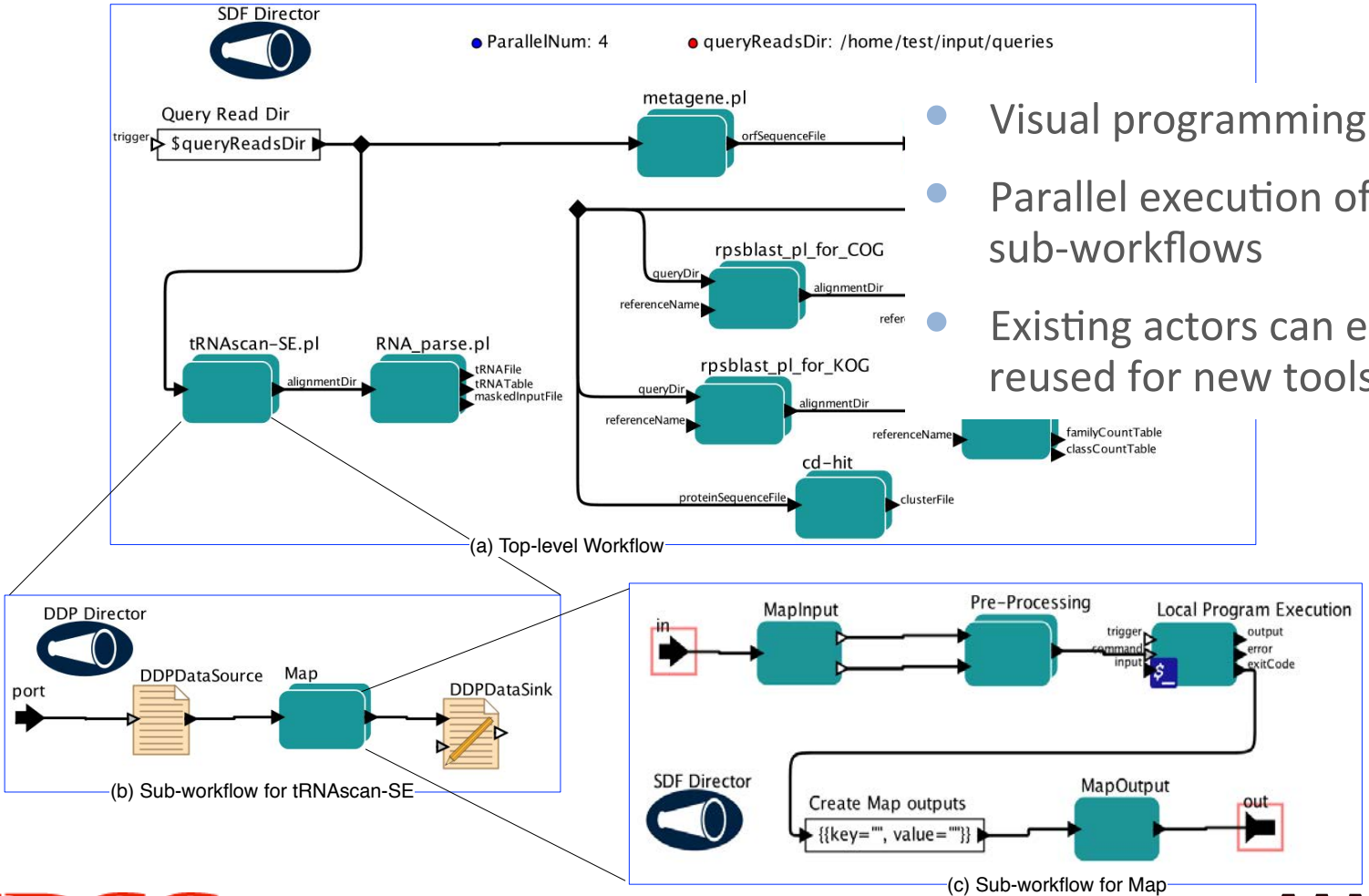
Support Big Data Patterns in Kepler

- We define a separate DDP (Distributed Data-Parallel) task/actor for each pattern
- These DDP actors partition input data and process each partition separately
- User-defined functions are described as sub-workflows of DDP actors
- DDP director: executes DDP workflows on top of Big Data engines

SDSC
SAN DIEGO SUPERCOMPUTER CENTER

WorDS Center

RAMMCAP in Kepler Workflow System

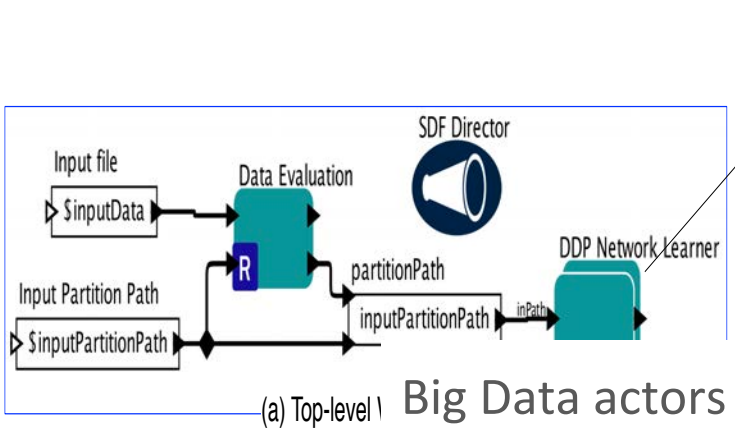


- Visual programming
- Parallel execution of the DDP sub-workflows
- Existing actors can easily reused for new tools

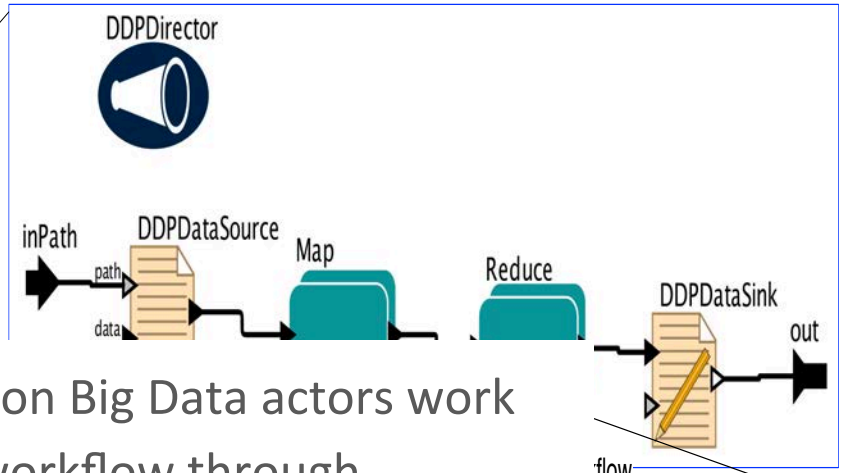
SDSC
SAN DIEGO SUPERCOMPUTER CENTER

WorDS Center

Kepler Workflow for Bayesian Network Learning Application

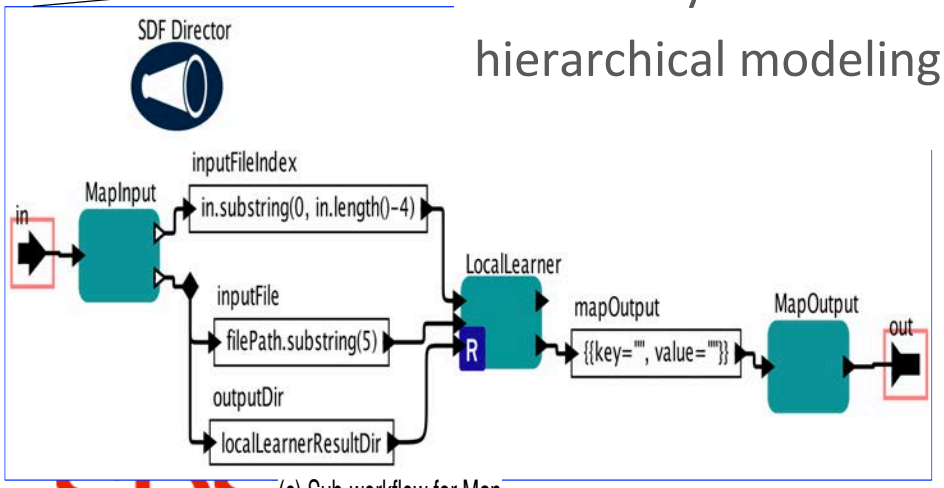


(a) Top-level

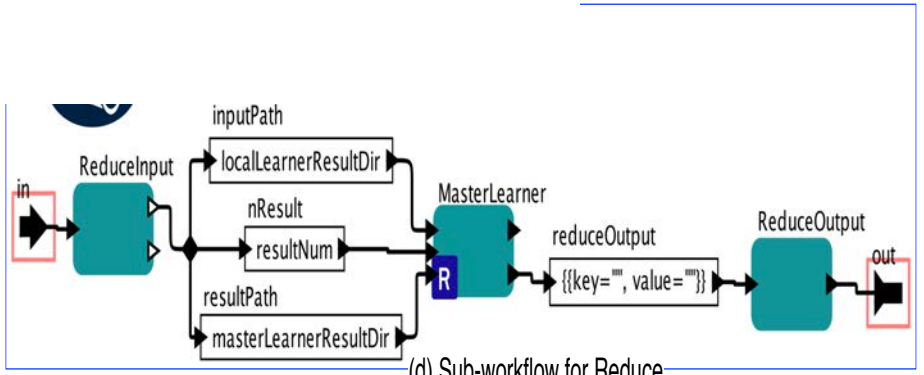


:flow

Big Data actors and non Big Data actors work seamlessly within a workflow through hierarchical modeling



(c) Sub-workflow for Map



(d) Sub-workflow for Reduce

SDSC
SAN DIEGO SUPERCOMPUTER CENTER

WorDS Center

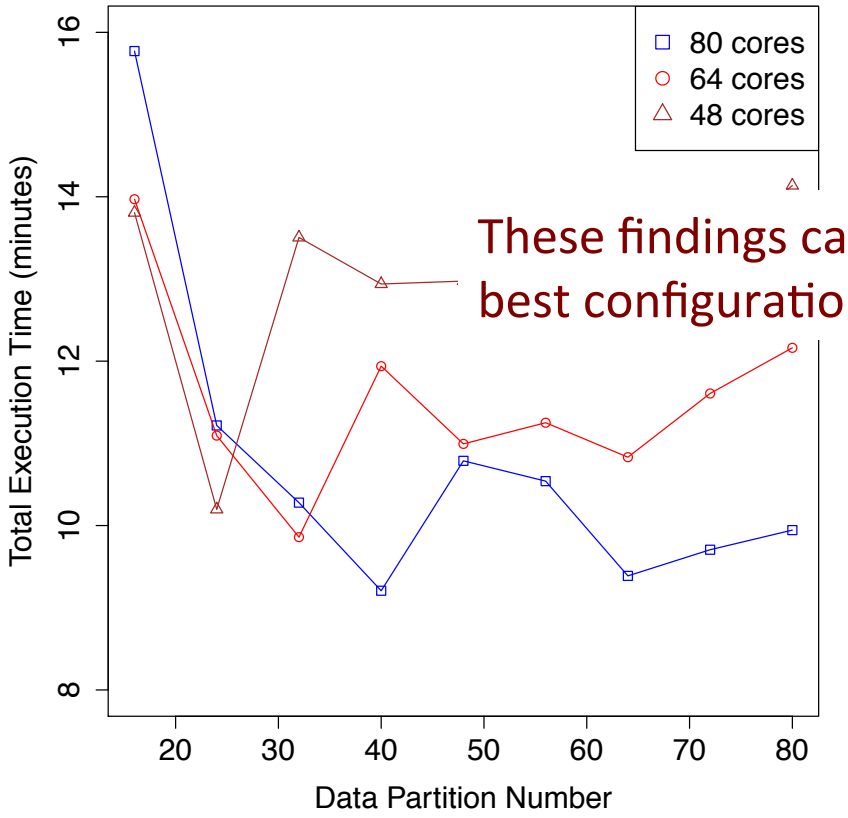
Legacy Tool Parallelization for Big Data

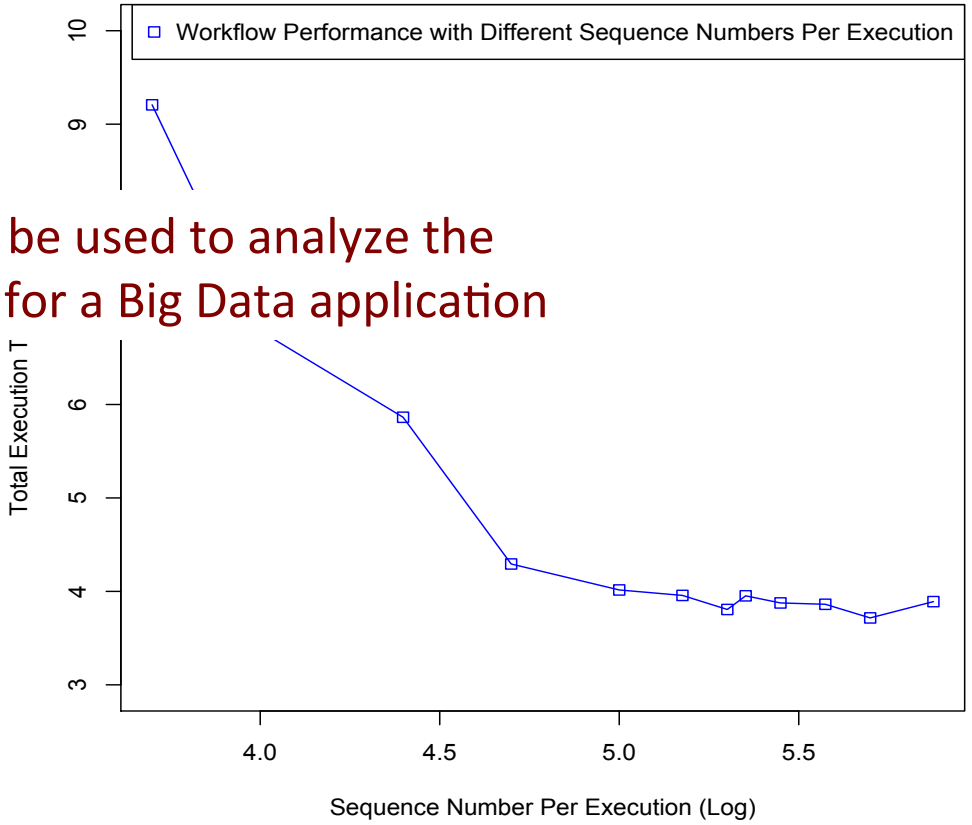
- Black-box approach (we use this approach)
  - Run a tool directly
  - Wrap the tool with Big Data techniques
  - Can quickly convert a tool into a parallelized one
- White-box approach
  - Investigate the source code of a legacy tool and try to re-implement it using Big Data techniques
  - Time-consuming
  - Often tightly-coupled with specific Big Data engine
  - Could find more parallel opportunities

Performance Factors of Legacy Tool Parallelization

- Data partitioning
  - By default, data is partitioned to 64 MB blocks in Hadoop
  - Load balancing is more important because legacy tool execution can take a long time even with small input
- Data size for each legacy tool execution
  - New Big Data analytics applications often try to process minimal input data for each execution
  - Because of legacy tool execution overhead, performance is better if each execution processes relatively large input

Experiments on Performance Factors





These findings can be used to analyze the best configuration for a Big Data application

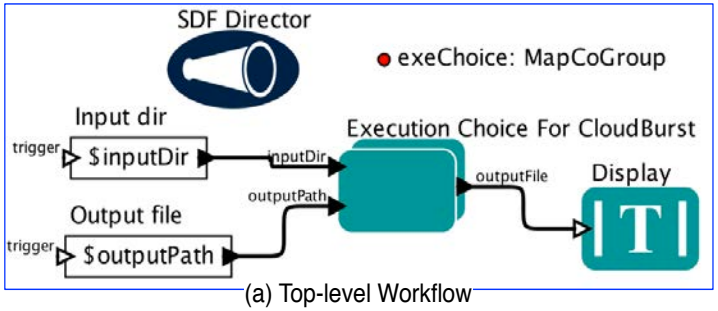
SDSC
SAN DIEGO SUPERCOMPUTER CENTER

WorDS Center

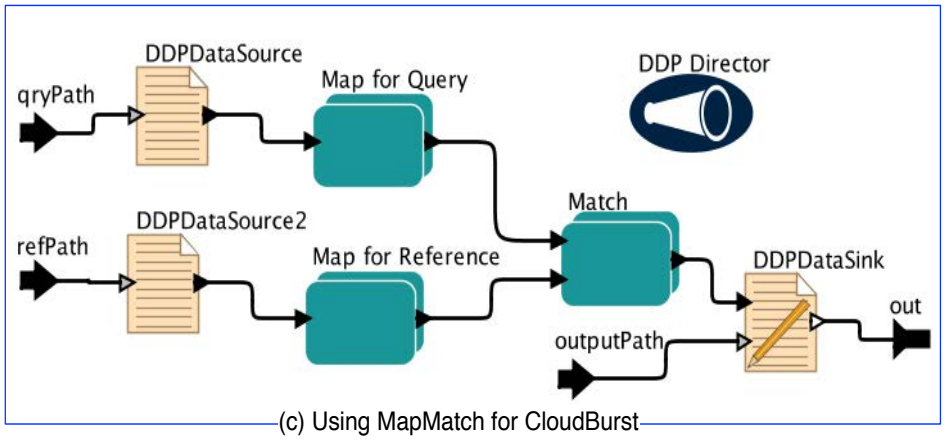
Big Data Pattern Selection

- A Big Data task could often be executed using different Big Data patterns
- CloudBurst is a parallel mapping tool in bioinformatics implemented using MapReduce
- We re-implement CloudBurst using MapCoGroup and MapMatch and find they are easier to build
- Our analysis and experiments show no pattern is always the best in terms of performance
- Performance depends on input data characteristics
  - The balance of the two input datasets
  - The sparseness of the values for each key

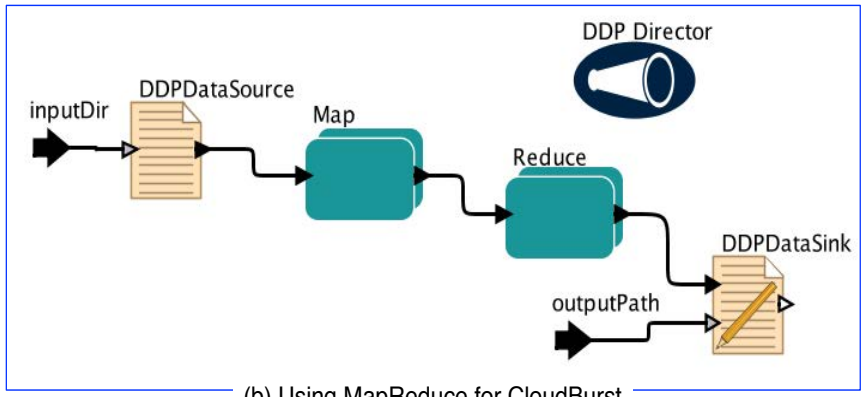
Kepler Workflow for CloudBurst Application



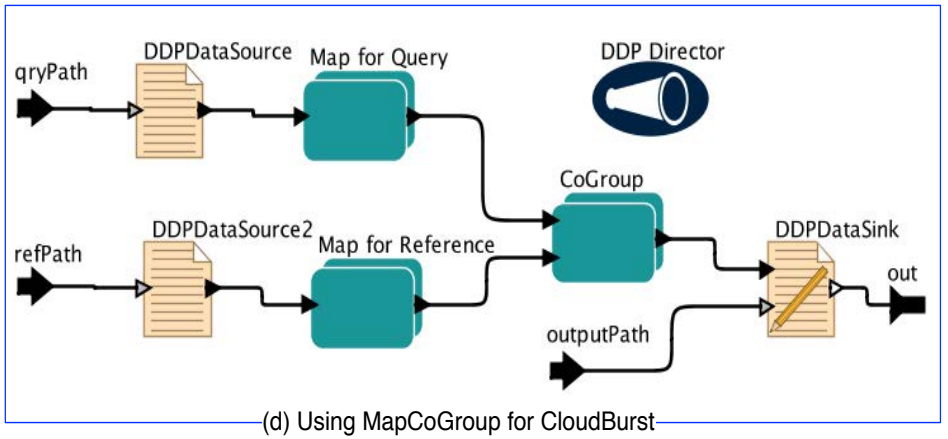
(a) Top-level Workflow



(c) Using MapMatch for CloudBurst



(b) Using MapReduce for CloudBurst

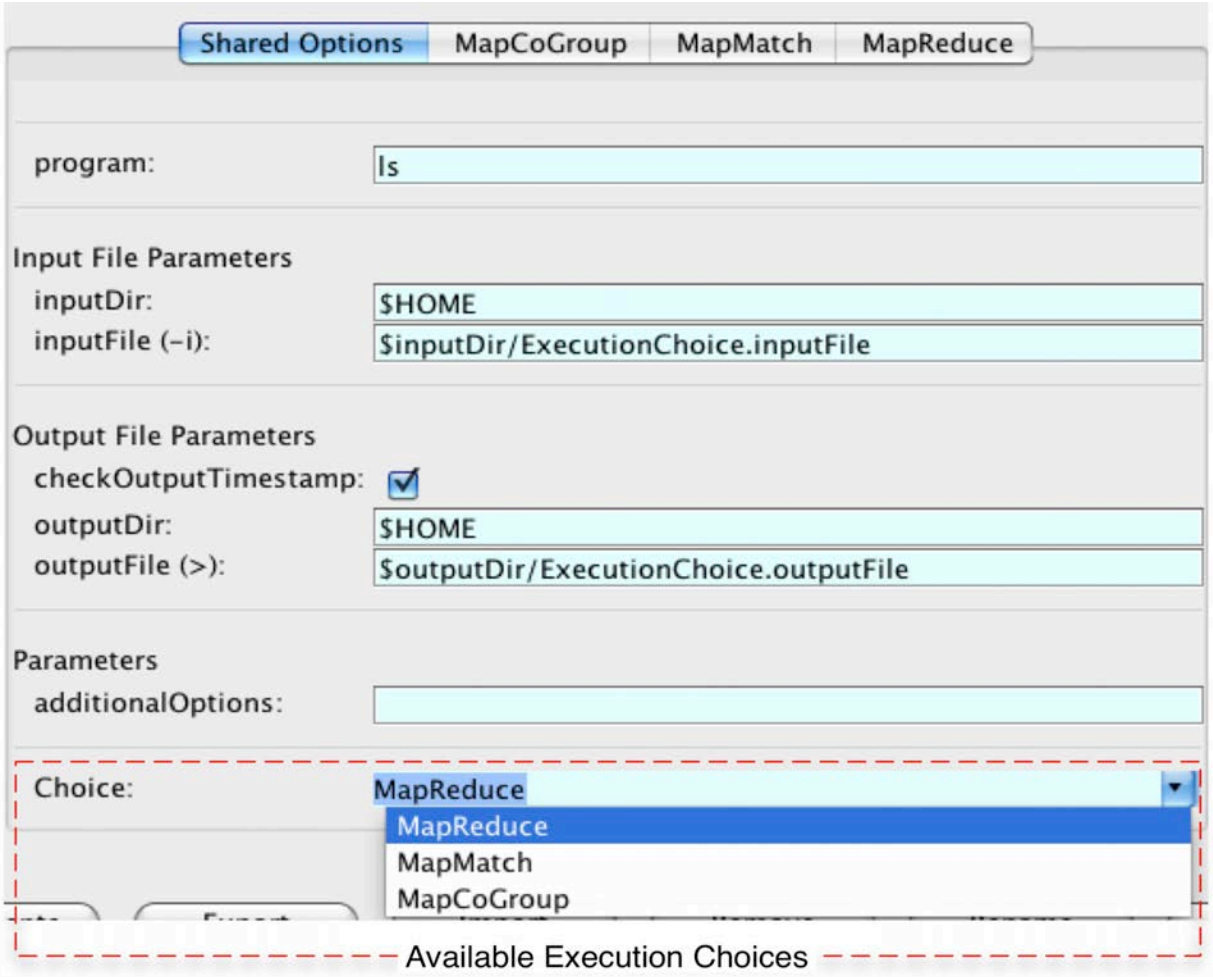


(d) Using MapCoGroup for CloudBurst

SDSC
SAN DIEGO SUPERCOMPUTER CENTER

WorDS Center

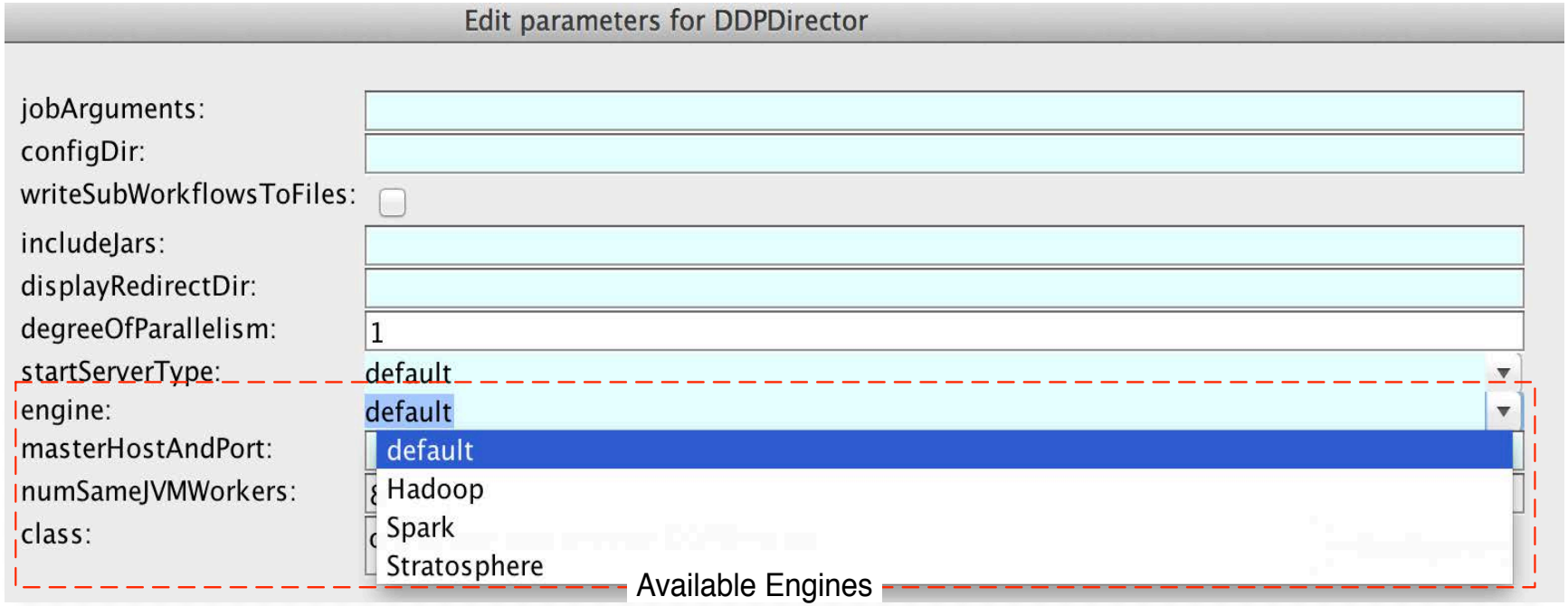
Execution Choices for CloudBurst Application



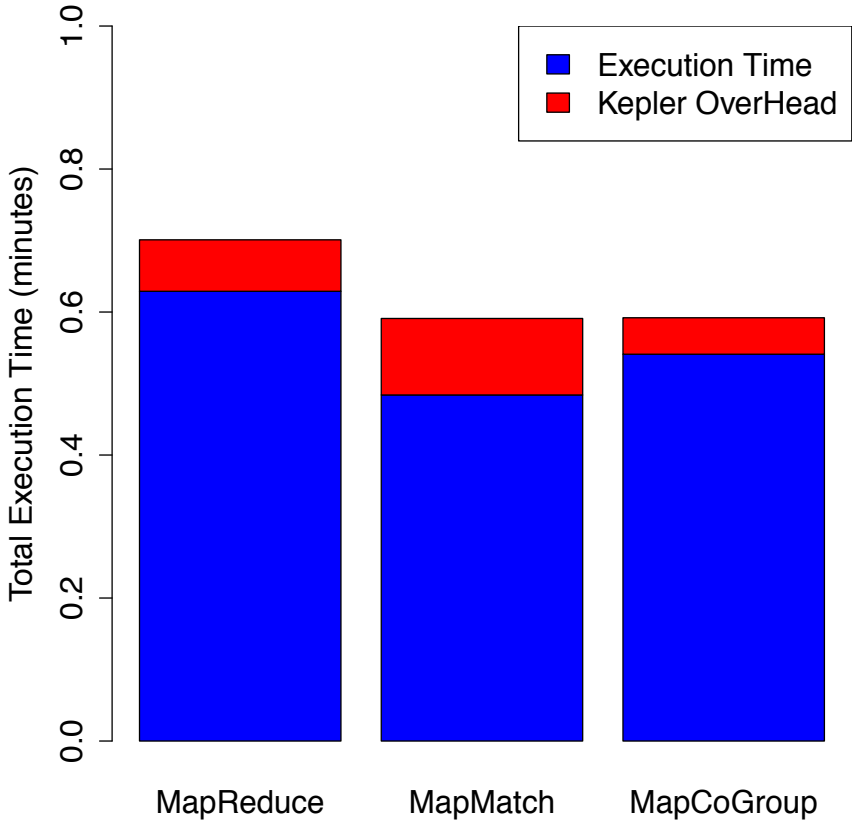
Big Data Process Execution

- Adaptability: Our DDP director can run the same process/workflow on different Big Data engines
- The director transforms workflow into jobs based on each Big Data engine's specification
- For Hadoop, CoGroup, Match and Cross patterns are not supported directly. The director converts them into MapReduce jobs
- Consecutive Map patterns are automatically merged before execution to improve performance

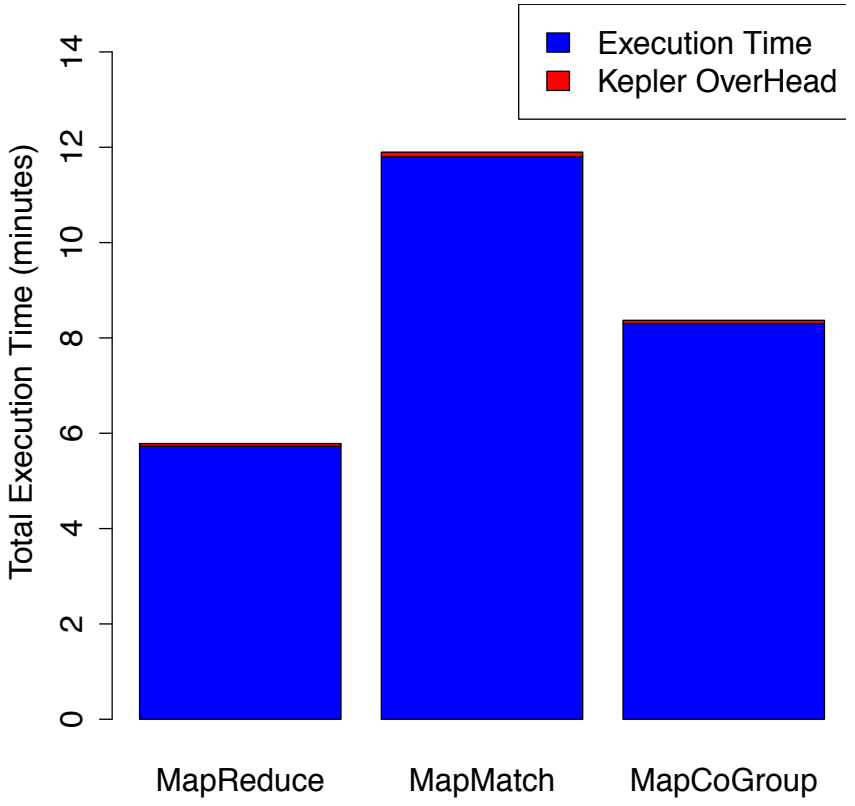
Engine Configuration of DDP Director



Overhead Experiments



Kepler overhead for CloudBurst application with small input data



Kepler overhead for CloudBurst application with large input data

SDSC
SAN DIEGO SUPERCOMPUTER CENTER

WorDS Center

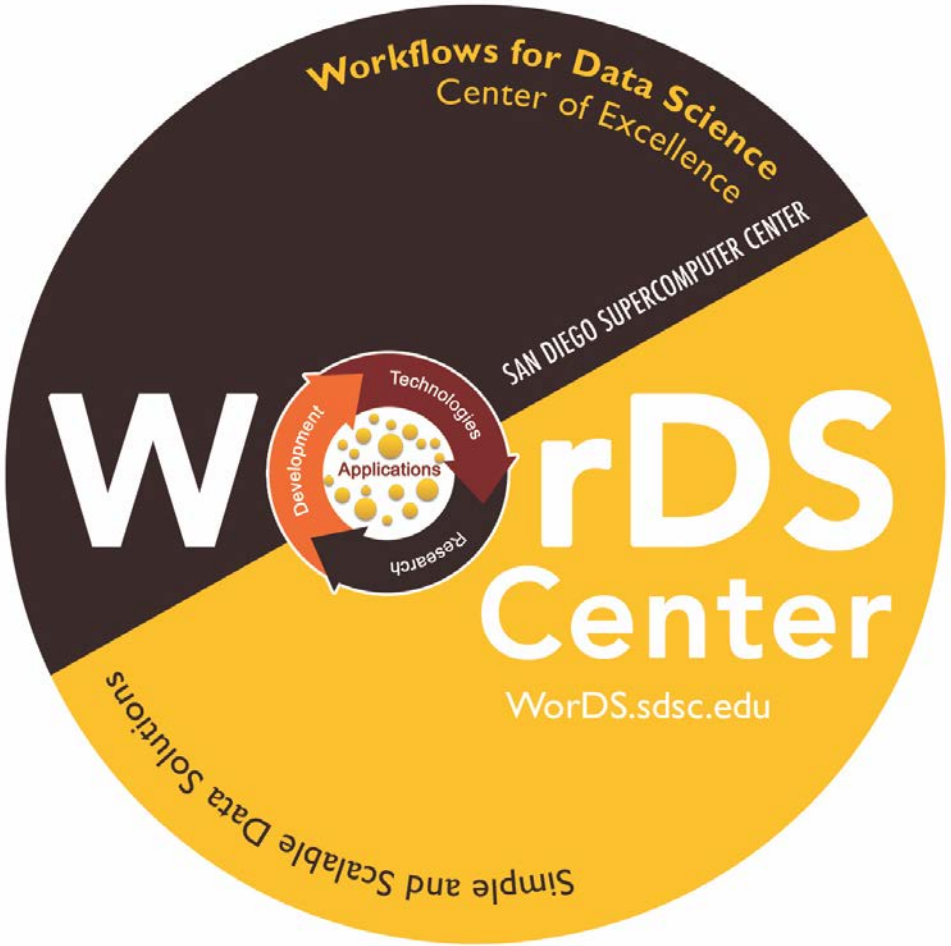
Conclusions

- Usability/programmability
  - Easy Big Data application construction through visual programing
  - The same Big Data application can execute on different Big Data engines by the DDP director
- Execution optimization
  - The findings on legacy tool parallelization and Big Data pattern selection can help optimal execution
  - The additional layer for workflow system brings minor overhead

More Research Challenges

- End-to-end performance prediction for Big Data applications/workflows (how long to run)
  - Knowledge based: Analyze performance using profiling techniques and dependency analysis
  - Data driven: Predict performance based on execution history (provenance) using machine learning techniques
- On demand resource provisioning and scheduling for Big Data applications (where and how to run)
  - Find the best resource allocation based on execution objectives and performance predictions
  - Find the best workflow and task configuration on the allocated resources

Questions?



WorDS Director: Ilkay Altintas, Ph.D.
Email: altintas@sdsc.edu

SDSC
SAN DIEGO SUPERCOMPUTER CENTER

WorDS Center