Pwrake

Distributed Workflow Engine for e-Science

Masahiro Tanaka
University of Tsukuba

Who am I

- Masahiro Tanaka
- NArray author
- majored in Astronomy
- Research fellow in Computer Science
  - at Center for Computational Sciences, University of Tsukuba
  - since 2009

CCS, U. Tsukuba

- Research Fields
  - **Computer Science:**
    - High Performance Computing
    - Computational Informatics
  - **Computational Science:**
    - Particle Physics, Astrophysics, Material Science, Life Science, Biology, Environmental Science

CCS operates SuperComputers

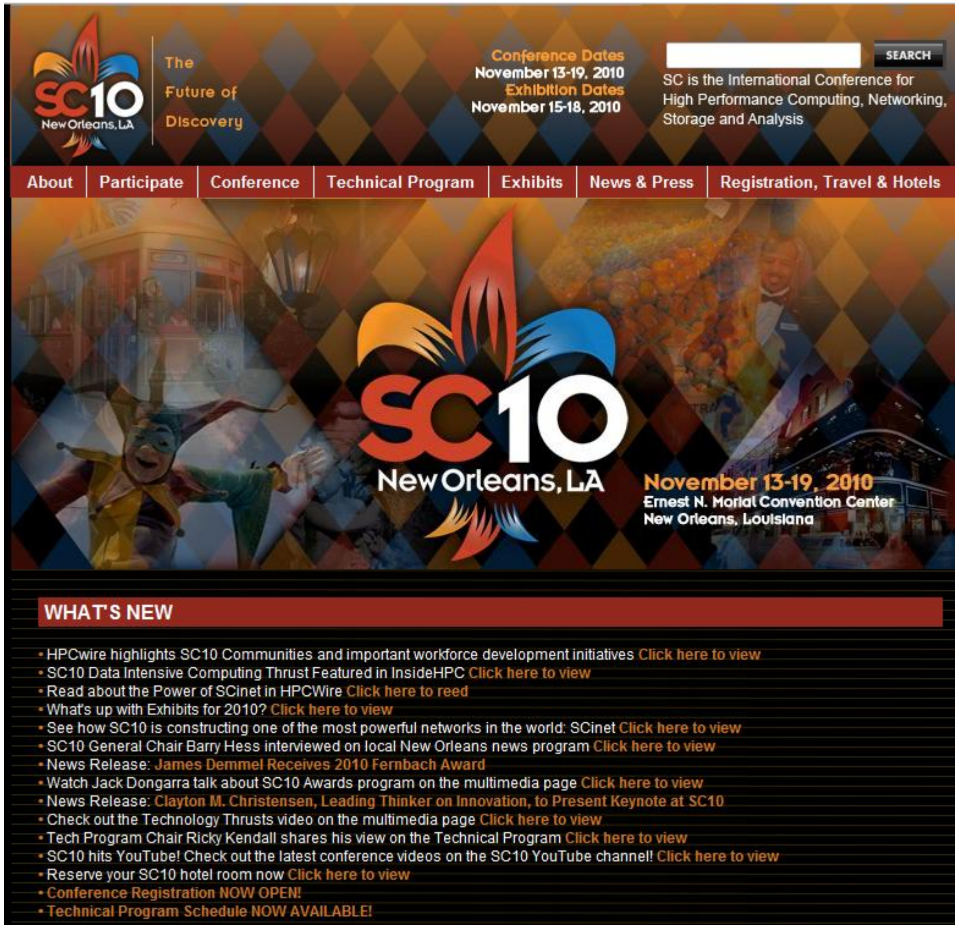
- **FIRST**
  - 512 cores+BladeGRAPE
  - 36 TFLOPS

- **PACS-CS**
  - 2,560 cores
  - 14.4 TFOPS

- **T2K Tsukuba**
  - 10,368 cores
  - 95 TFLOPS

SC10

- Conference on SuperComputer
- > 10,000 participants



Conference Dates
November 13-19, 2010
Exhibition Dates
November 15-18, 2010

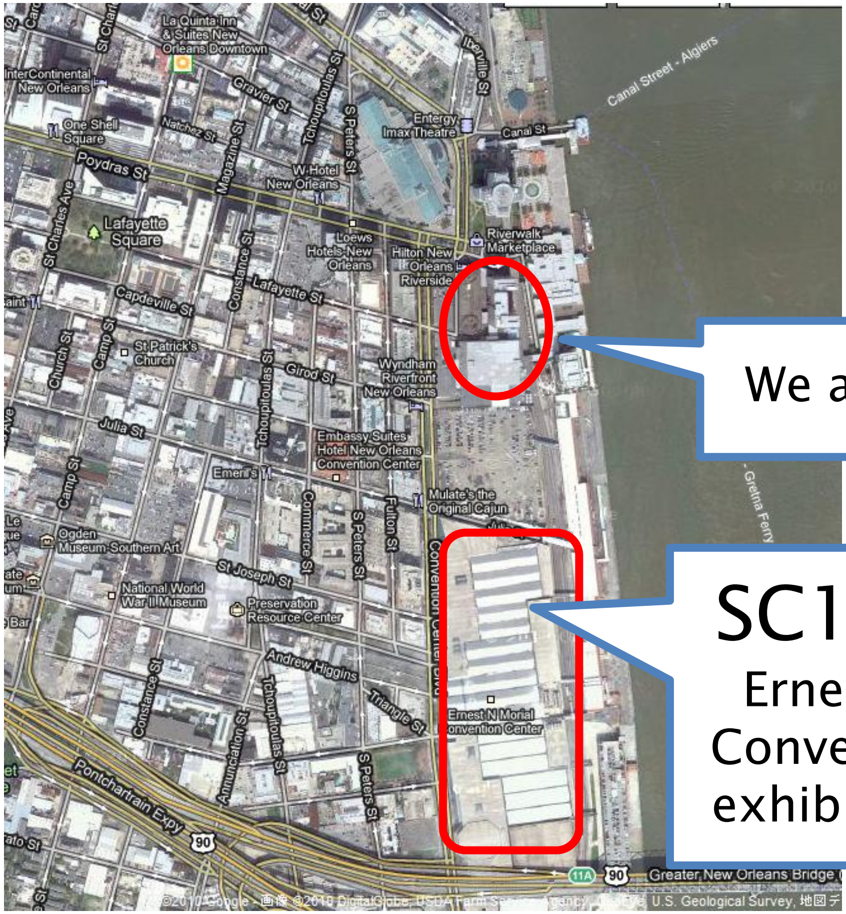
SC is the International Conference for High Performance Computing, Networking, Storage and Analysis

November 13-19, 2010
Ernest N. Morial Convention Center
New Orleans, Louisiana

WHAT'S NEW

- HPCwire highlights SC10 Communities and important workforce development initiatives Click here to view
- SC10 Data Intensive Computing Thrust Featured in InsideHPC Click here to view
- Read about the Power of SCinet in HPCWire Click here to reed
- What's up with Exhibits for 2010? Click here to view
- See how SC10 is constructing one of the most powerful networks in the world: SCinet Click here to view
- SC10 General Chair Barry Hess interviewed on local New Orleans news program Click here to view
- News Release: James Demmel Receives 2010 Fernbach Award
- Watch Jack Dongarra talk about SC10 Awards program on the multimedia page Click here to view
- News Release: Clayton M. Christensen, Leading Thinker on Innovation, to Present Keynote at SC10
- Check out the Technology Thrusts video on the multimedia page Click here to view
- Tech Program Chair Ricky Kendall shares his view on the Technical Program Click here to view
- SC10 hits YouTube! Check out the latest conference videos on the SC10 YouTube channel! Click here to view
- Reserve your SC10 hotel room now Click here to view
- Conference Registration NOW OPEN!
- Technical Program Schedule NOW AVAILABLE!

SC10 is held in next week



We are here

SC10 venue

Ernest N. Morial Convention Center exhibit Nov 15-18

CCS has a booth at SC10

- CCS booth at SC09

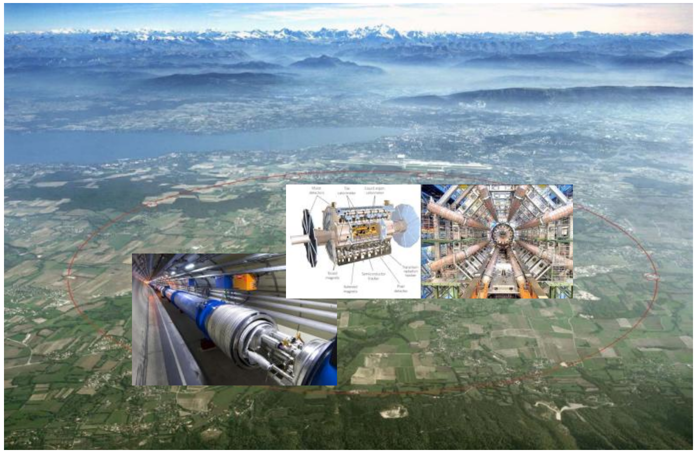
Today's Topic

- Pwrake : a Distributed Workflow Engine for e-Science

Introduction

Science conducted under international collaboration

LHC
Particle Accelerator

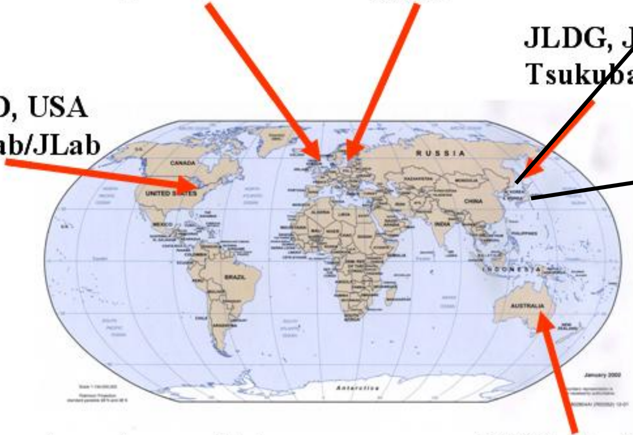


ALMA
Radio Observatory

Geographically Distributed Computer Resources

ILDG and JLDG : Sharing QCD Simulation data

- A Grid of Grids, or aggregation of five regional grids.
- JLDG is the ILDG Japan grid (gateway at CCS, University of Tsukuba).



UKQCD (QCDgrid/DiGS), UK, Edinburgh

LatFOR, Germany/France/Italy DESY

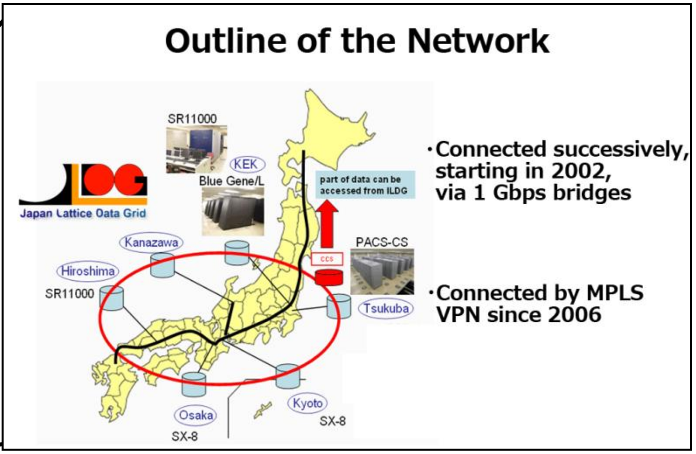
JLDG, Japan Tsukuba

USQCD, USA Fermilab/JLab

CSSM, Australia Adelaide

ILDG

http://www.lqcd.org/ildg



Outline of the Network

- Connected successively, starting in 2002, via 1 Gbps bridges
- Connected by MPLS VPN since 2006

http://www.sinet.ad.jp/case-examples/tsukuba

e-Science

- Computationally intensive science that is carried out in highly distributed network environments,

or

- Science that uses immense data sets that require grid computing
  - (Wikipedia).

e-Science

- The term was created by John Taylor,
  - Director General of the United Kingdom's Office of Science and Technology
  - in 1999

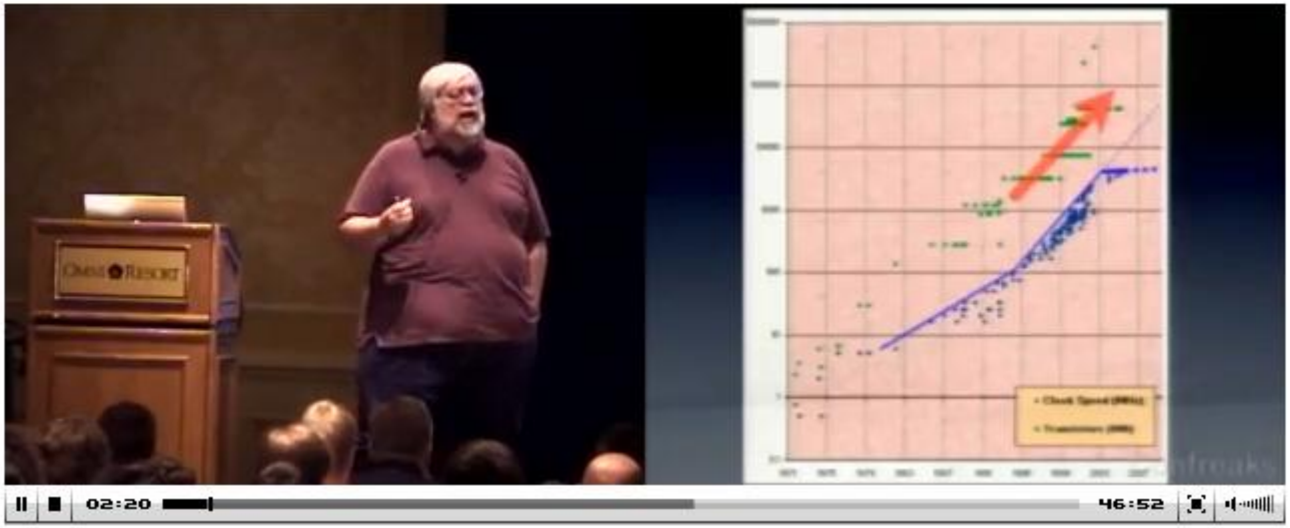
Distributed Computing

- is a key issue for e-Science.

End of Moore’s law

- Performance of single core does no more increase.

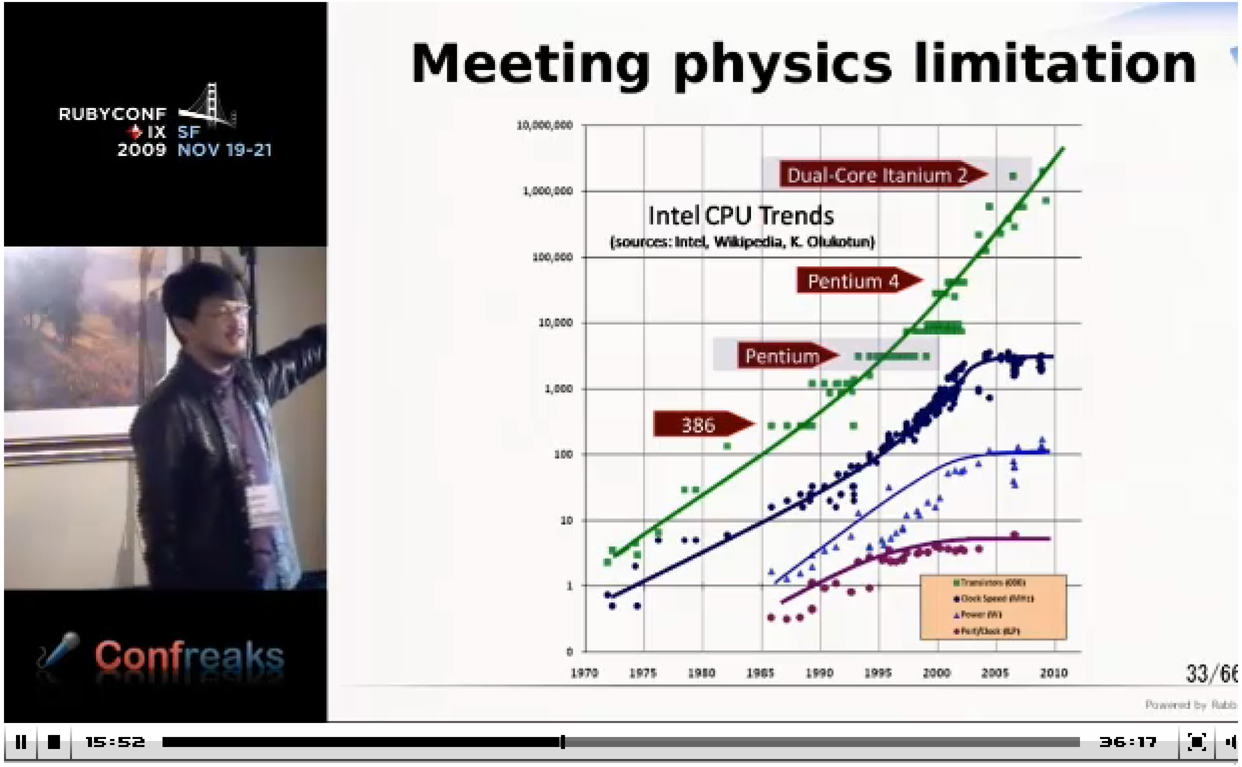
RubyConf 2008



What All Rubyist Should Know About Threads

Jim Weirich

Ruby Conference 2009

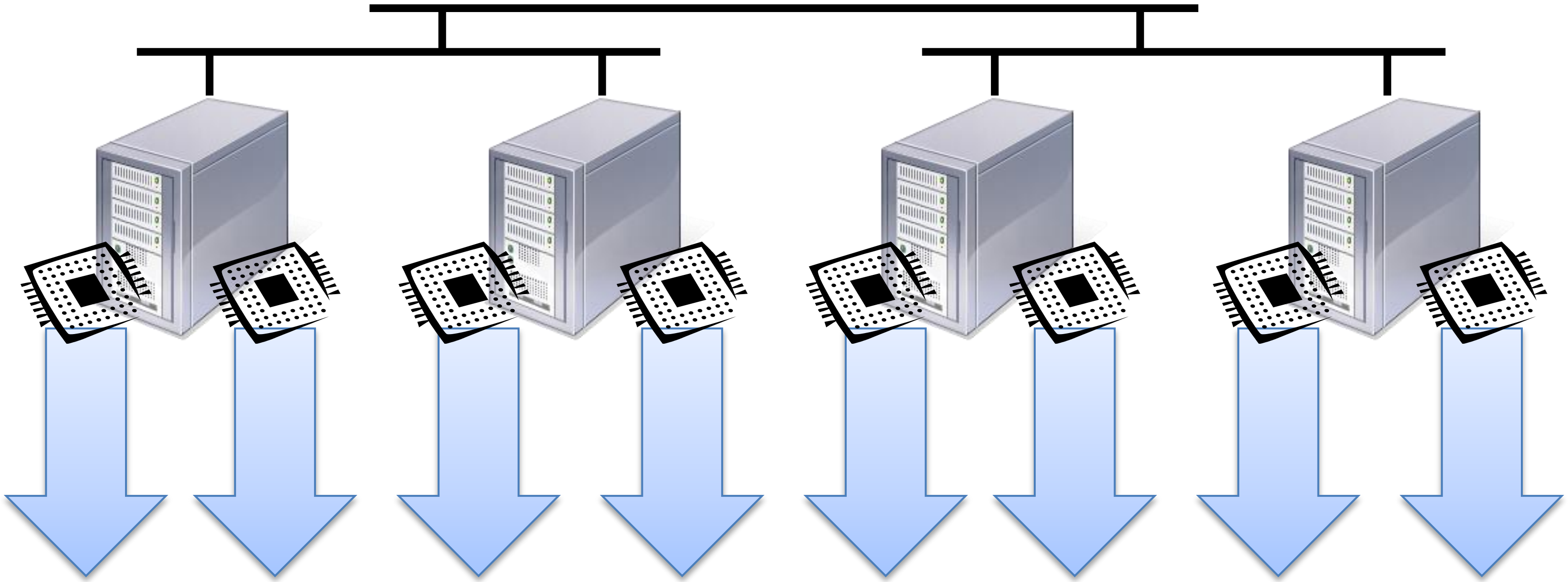


Keynote Address

Yukihiro 'Matz' Matsumoto

Using Multi-Core

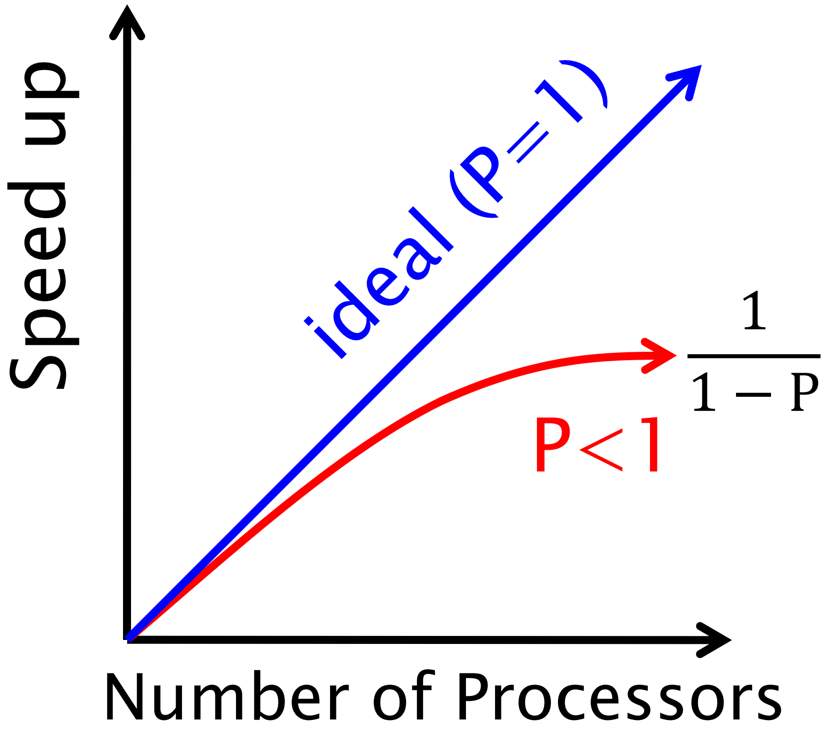
- Parallelize your program
- Scalability is an key issue



Amdahl's law

- P : parallelizable
- 1−P : sequential
- N : # of processors
- Speed-up formula :

$$\frac{1}{1-\mathrm{P}+\mathrm{P}/\mathrm{N}}$$



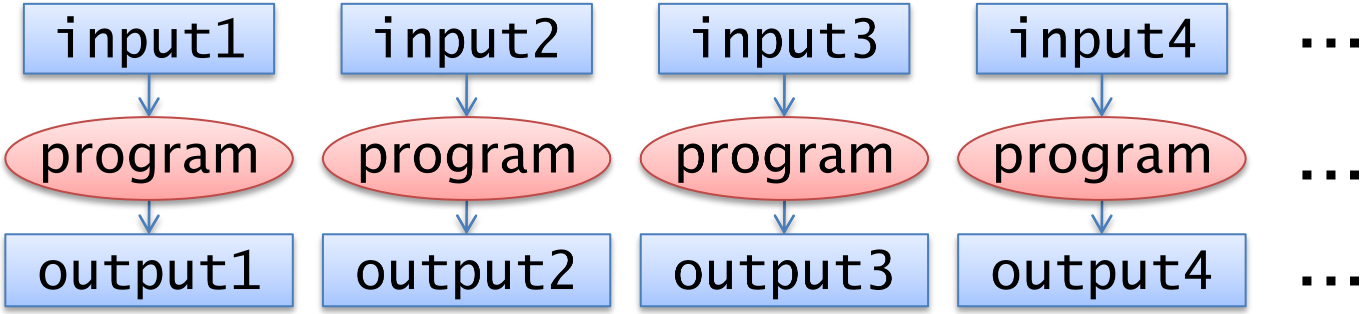
Parallel Programming models

- MapReduce
- MPI
- OpenMP
- thread
- Parallel programming languages

- process

Parallelize Processes

- Independent processes can be parallelized
- Without parallel programming
- Workflow System is required

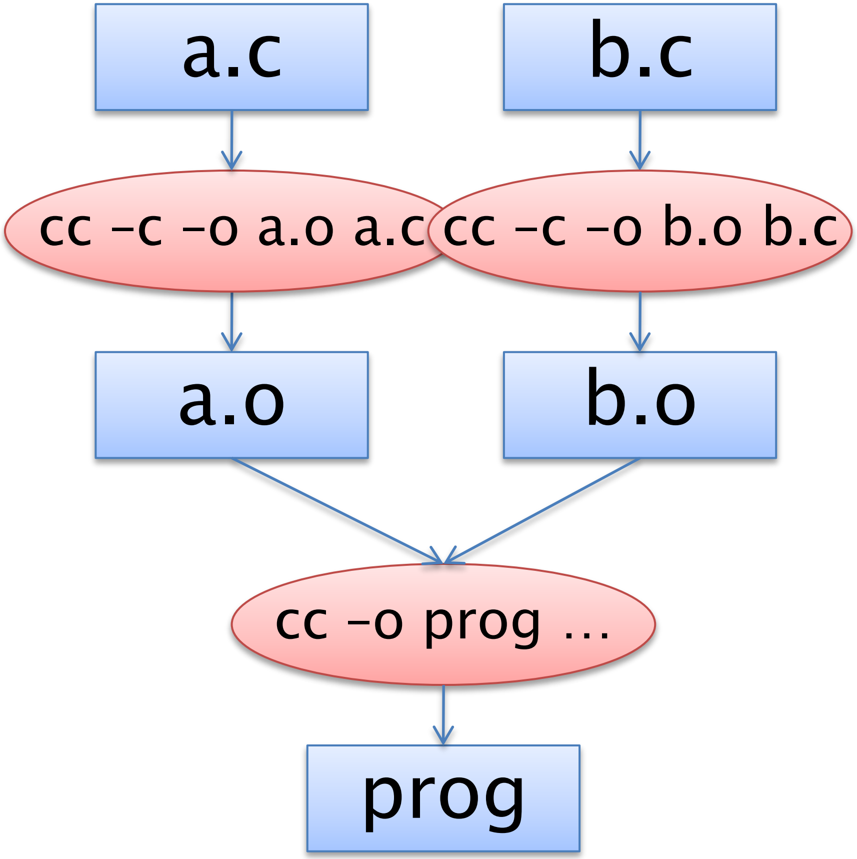


Scientific Workflow

Scientific Workflow

- Description of procedures

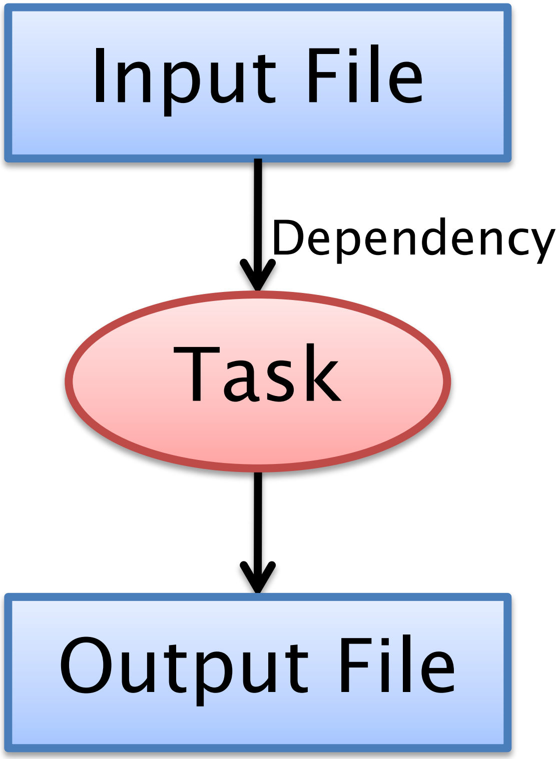
- It is like building a program



Graph representation

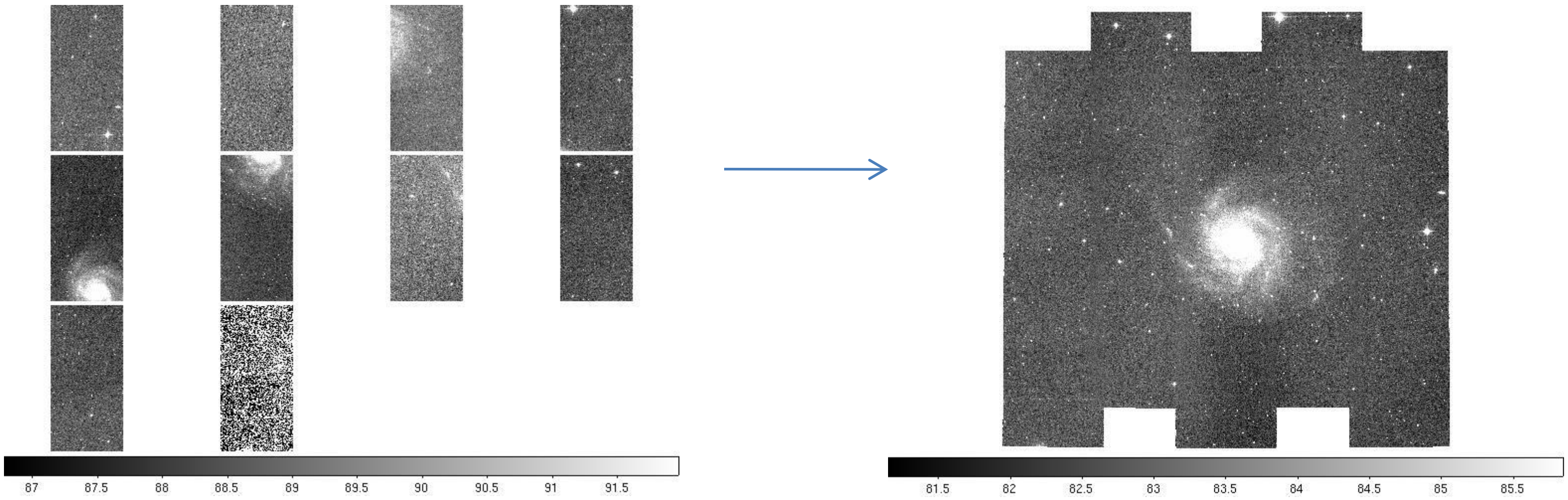
- Task: Ellipse Node
- File: Rectangle Node
- Dependency: Edge

- **DAG**
  - Directed Acyclic Graph

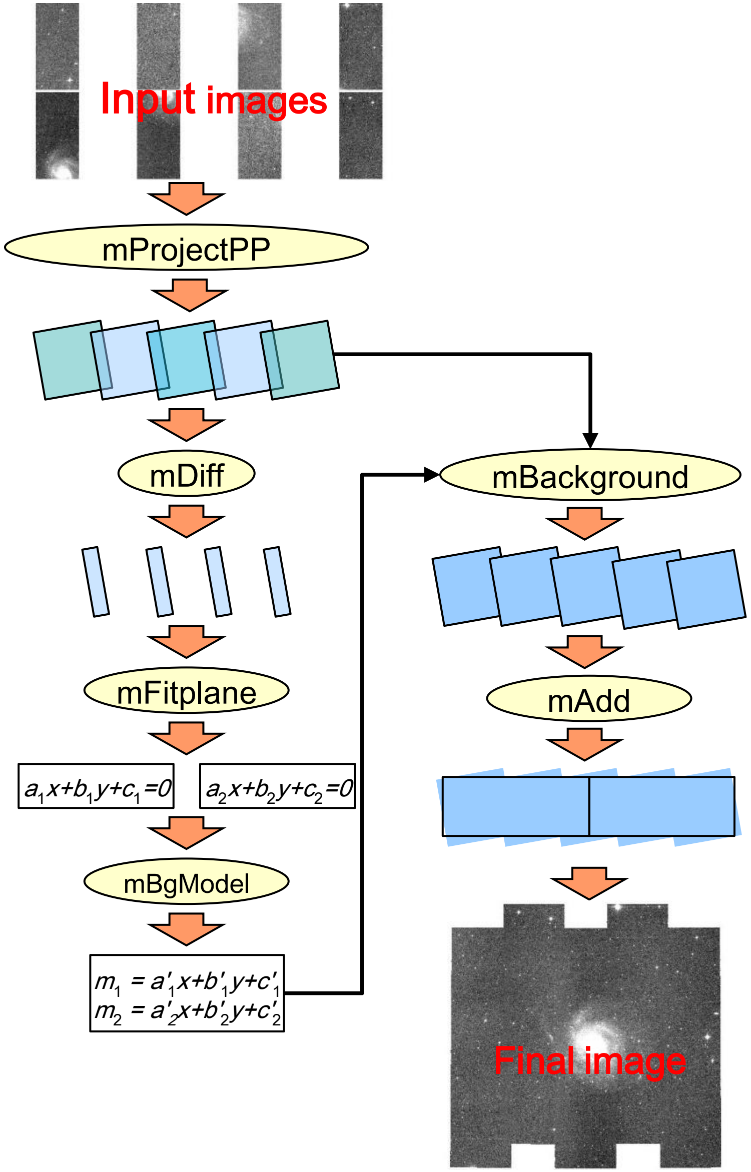


Example of Scientific Workflow

- Montage
  - software for producing a custom mosaic image from multiple shots of images.
  - http://montage.ipac.caltech.edu/

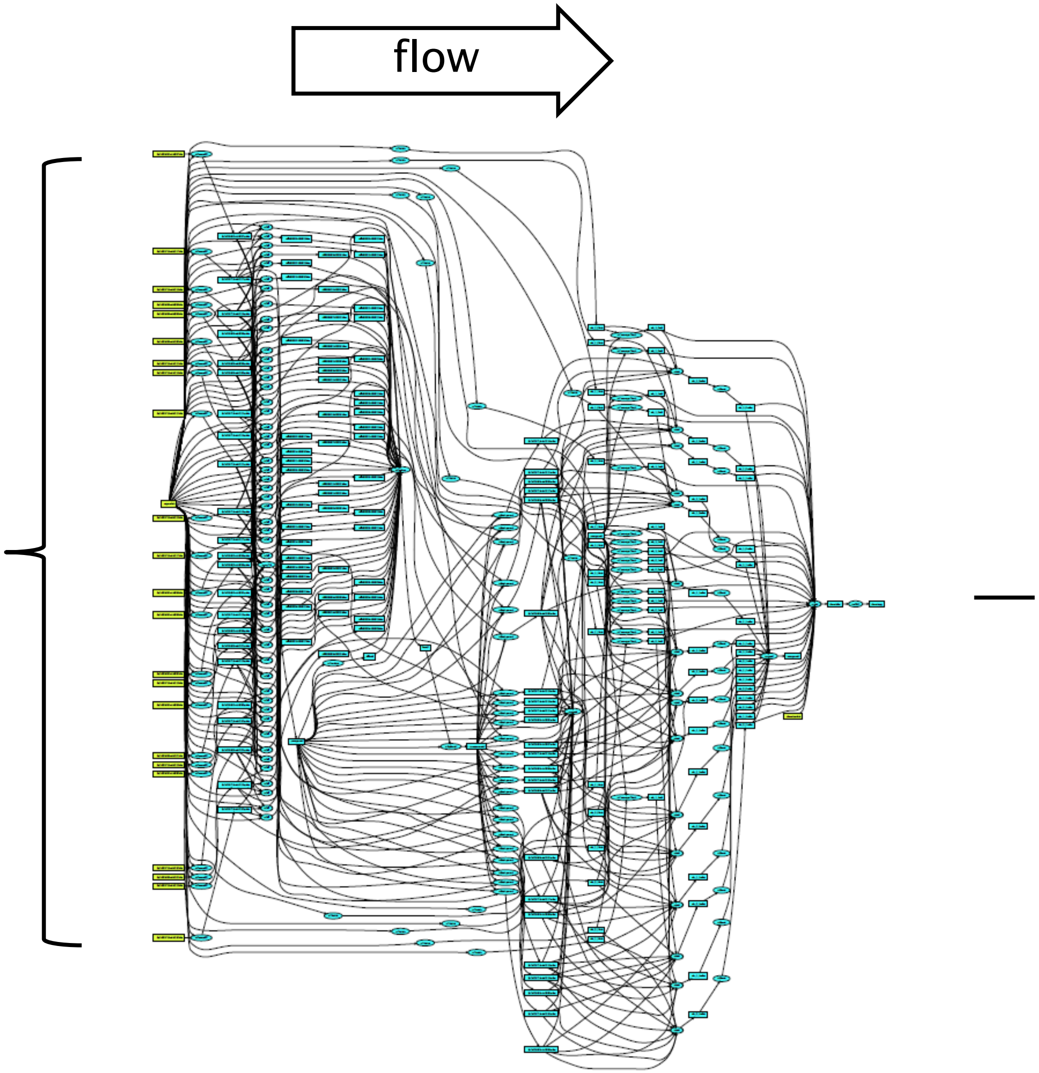


Montage Workflow



- Tasks:
  - Projection
  - Brightness correction
  - Coadding
- 1 image : 1 process

Workflow Graph of Montage



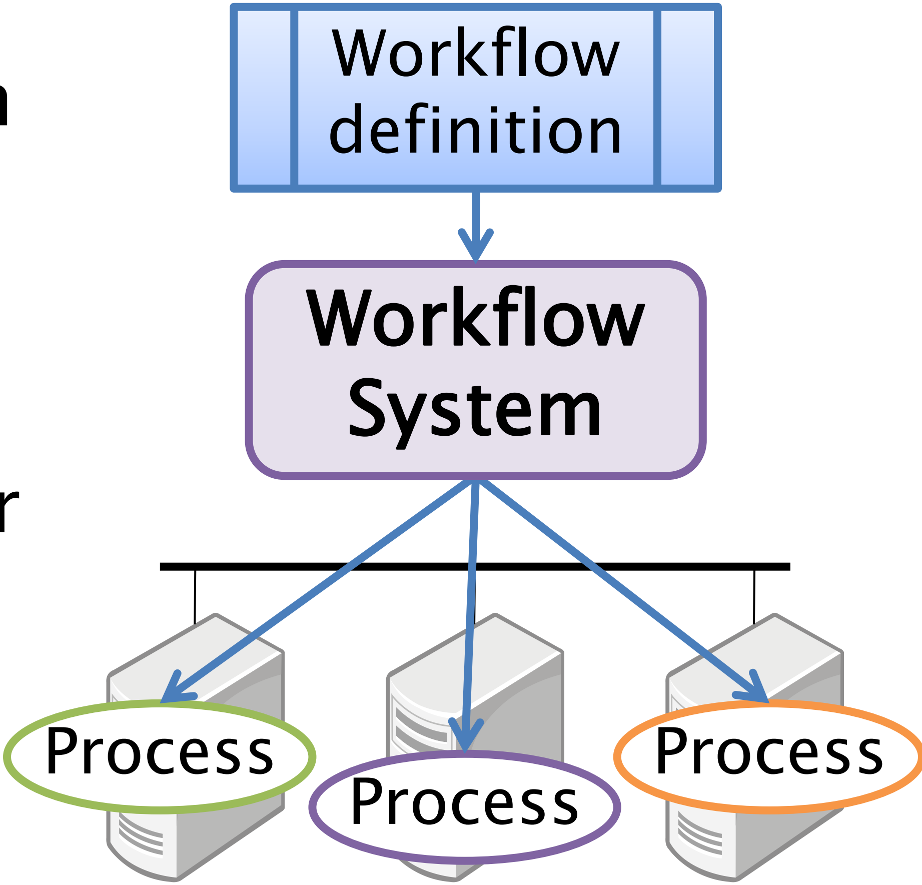
flow

Input files

Output file

Workflow System

- invoke task based on dependency
- assign a task to an available computer
- parallel execution for independent tasks



Scientific Workflow Systems

- for Grid Computing

- DAGMan
- Pegasus
- Triana
- ICENI
- Taverna
- GrADS
- GridFlow
- UNICORE
- Globus workflow
- Askalan
- Karajan
- Kepler

from “A Taxonomy of Scientific
Workflow Systems for Grid Computing”
Jia Yu and Rajkumar Buyya (2005)

Language for Scientific Workflow

- Define DAG in XML
  - Human cannot write complex XML.
  - Need to write a program to generate XML

DAG XML

```
<adag xmlns="http://www.griphyn.org/chimera/DAX"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://www.griphyn.org/chimera/DAX
       http://www.griphyn.org/chimera/dax-1.8.xsd"
     count="1" index="0" name="test">
<filename file="2mass-atlas-981204n-j0160056.fits"
       link="input"/>
 ...
 <job id="ID000001" name="mProject" version="3.0" level="11" dv-
       name="mProject1" dv-version="1.0">
   <argument>
     <filename file="2mass-atlas-981204n-j0160056.fits"/>
     <filename file="p2mass-atlas-981204n-j0160056.fits"/>
     <filename file="templateTMP_AAAaaa01.hdr"/>
   </argument>
   <uses file="2mass-atlas-981204n-j0160056.fits" link="input"
       dontRegister="false" dontTransfer="false"/>
   <uses file="p2mass-atlas-981204n-j0160056.fits" link="output"
       dontRegister="true" dontTransfer="true"
       temporaryHint="tmp"/>
   <uses file="p2mass-atlas-981204n-j0160056_area.fits"
       link="output" dontRegister="true" dontTransfer="true"
       temporaryHint="tmp"/>
   <uses file="templateTMP_AAAaaa01.hdr" link="input"
       dontRegister="false" dontTransfer="false"/>
 </job>
 ...
 <child ref="ID003006">
   <parent ref="ID000001"/>
   <parent ref="ID000006"/>
 </child>
```

Make

- DSL to define task dependency
- Rule
  - define multiple tasks at once
  - avoid redundancy
- Skip finished tasks
  - based on timestamp of file

GXP

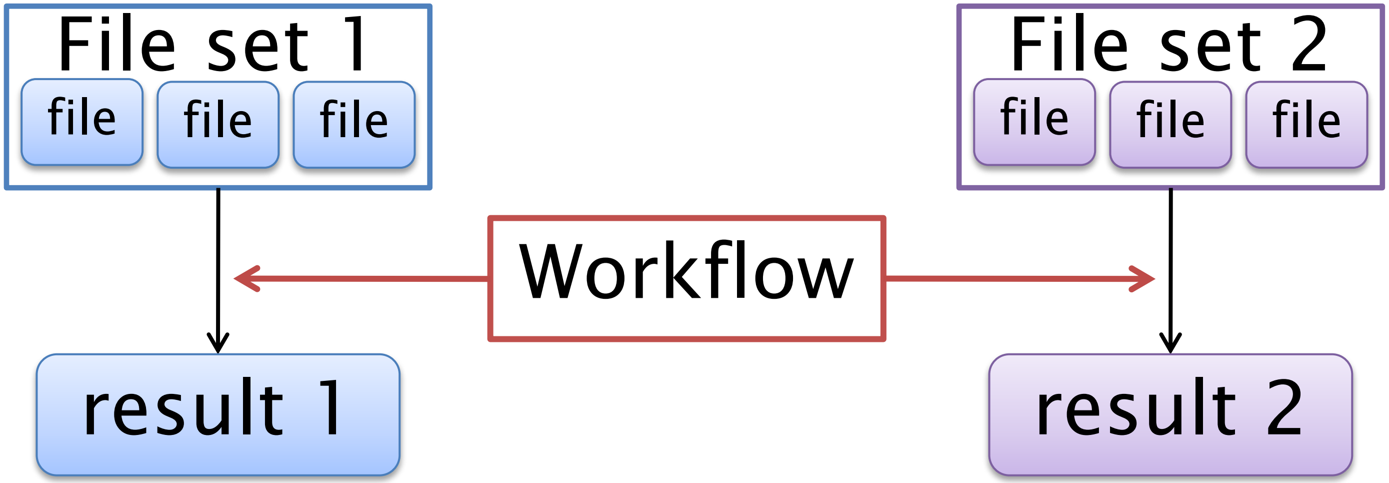
- Grid Explorer : Grid and Cluster shell
  - http://www.logos.ic.i.u-tokyo.ac.jp/gxp/
  - written in Python.
- GXP Make
  - GNU Make-based workflow system
  - Distributed & Parallel execution

Make is a build tool

- Makefile
  - same input files
  - same tasks
  - executed repeatedly
- Scientific Workflows have different aspects.

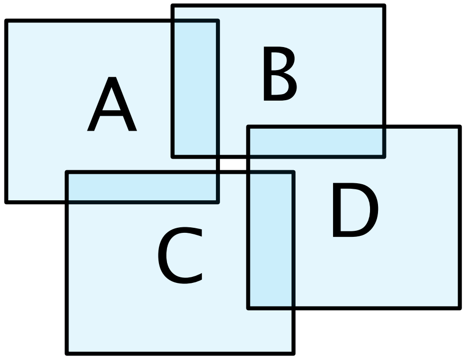
Aspect of Scientific Workflow (1/3)

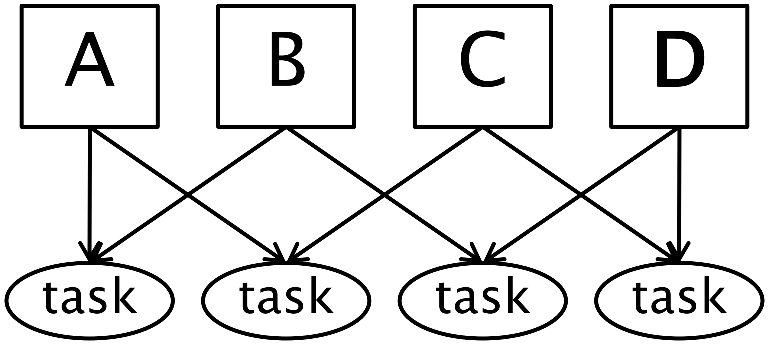
- Same workflow for different files
  - “`rule`” may solve, but is not enough.



Aspect of Scientific Workflow (2/3)

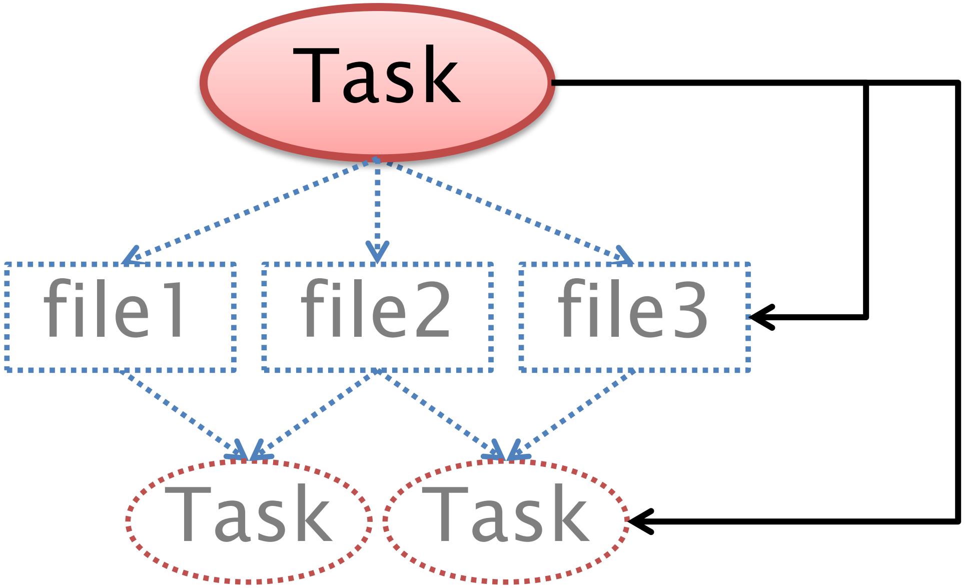
- Task dependencies rely on :
  - Not only file name
  - Parameters, e.g. Geometry





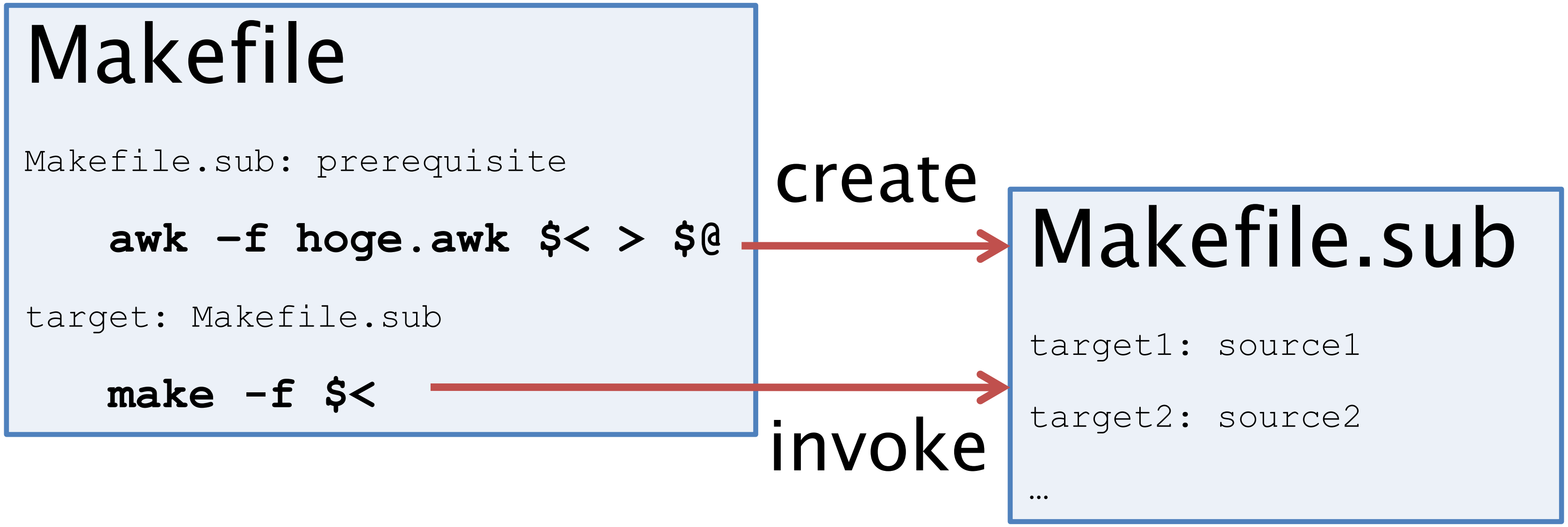
Aspect of Scientific Workflow (3/3)

- Entire workflow is unknown at first
- Result of a task affect
  - Output files
  - Afterward tasks



Dynamic task definition in Makefile

- create Makefile during Make execution
- tricky way



Makefile

```
Makefile.sub: prerequisite
    awk -f hoge.awk $< > $@
target: Makefile.sub
    make -f $<
```

Makefile.sub

```
target1: source1
target2: source2
…
```

- Scientific workflow requires powerful and flexible definition language.
- You probably know the solution.
  - What is it?

Rake

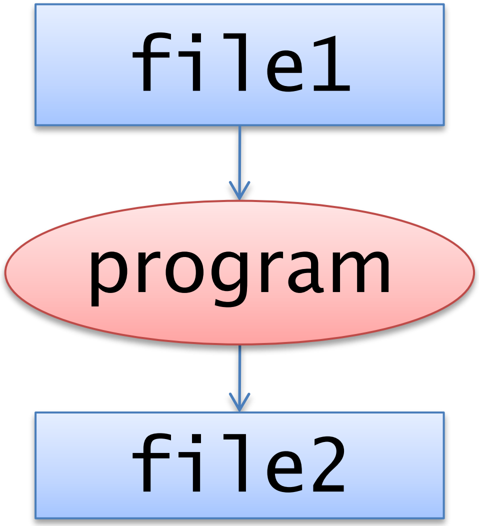
as a scientific workflow language

Rake

- Build tool
- Internal DSL
- Programming power of Ruby

Rakefile

```
file "file2" => "file1" do
  sh "program file1 > file2"
end
```

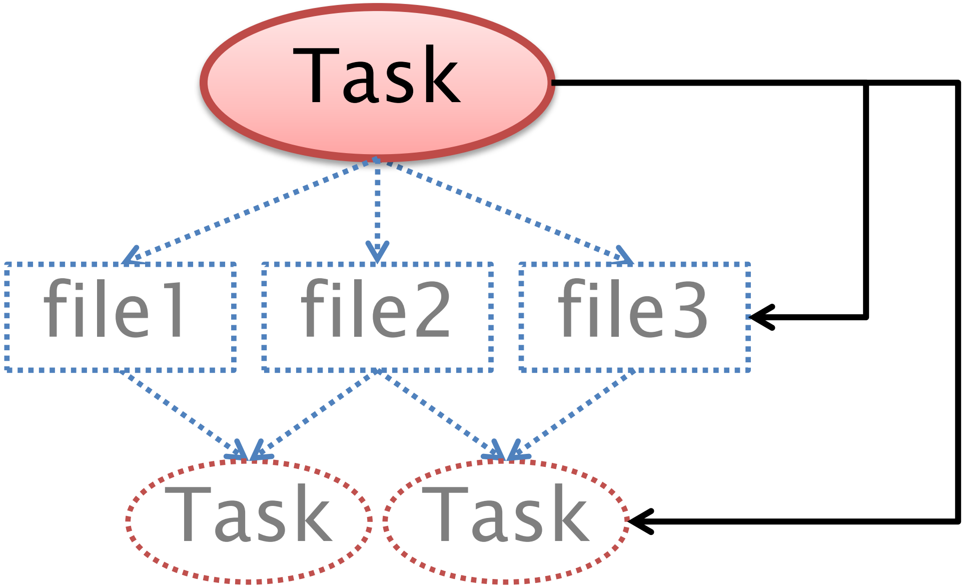


Define Tasks in Loop

```
for x in LIST
  file x[1] => x[0] do |t|
    sh "your_program …"
  end
end
```

Dynamic Task Definition

- How do your write it with Rake?



Task Definition in Task Action (Fail)

```
task :A do
  task :B do
    puts “B”
  end
end
task :default => :A
```

- No task depends on Task B

Task Definition in Task Action (Success)

```
task :A do
  b = task :B do
    puts “B”
  end
  b.invoke
end
task :default => :A
```

- Rake::Task#invoke
- Invoke Task B immediately after definition

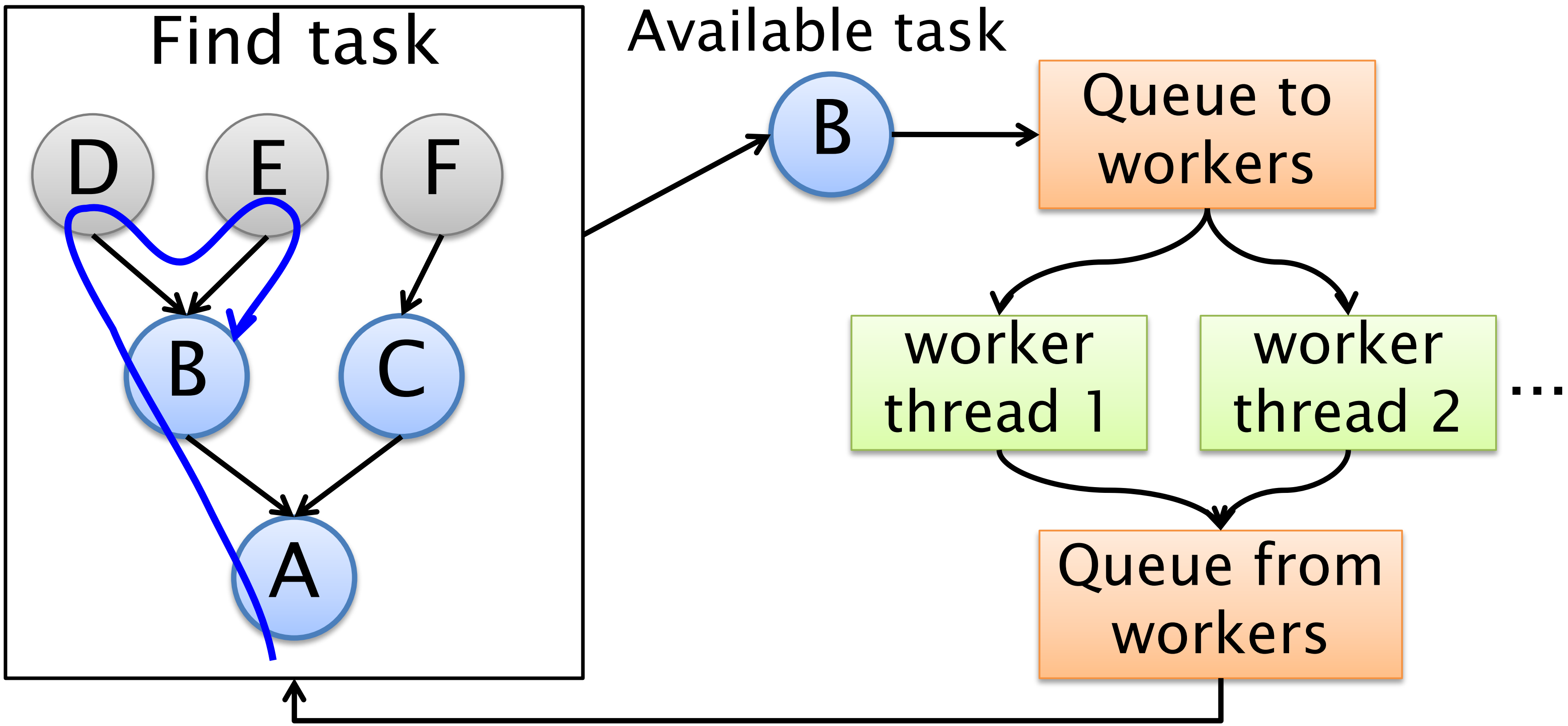
Parallelism in Rake

- **`multitask`**
  - Rake built-in feature
  - Parallelize prerequisite tasks of multitask
  - Ruby thread
- Problem
  - No control for the number of thread.
  - All the prerequisite tasks are invoked at the same time.

dRake

- http://drake.rubyforge.org/
- Specify the number of threads
- All the independent task are automatically parallelized.
    - multitask is not necessary

Design of dRake



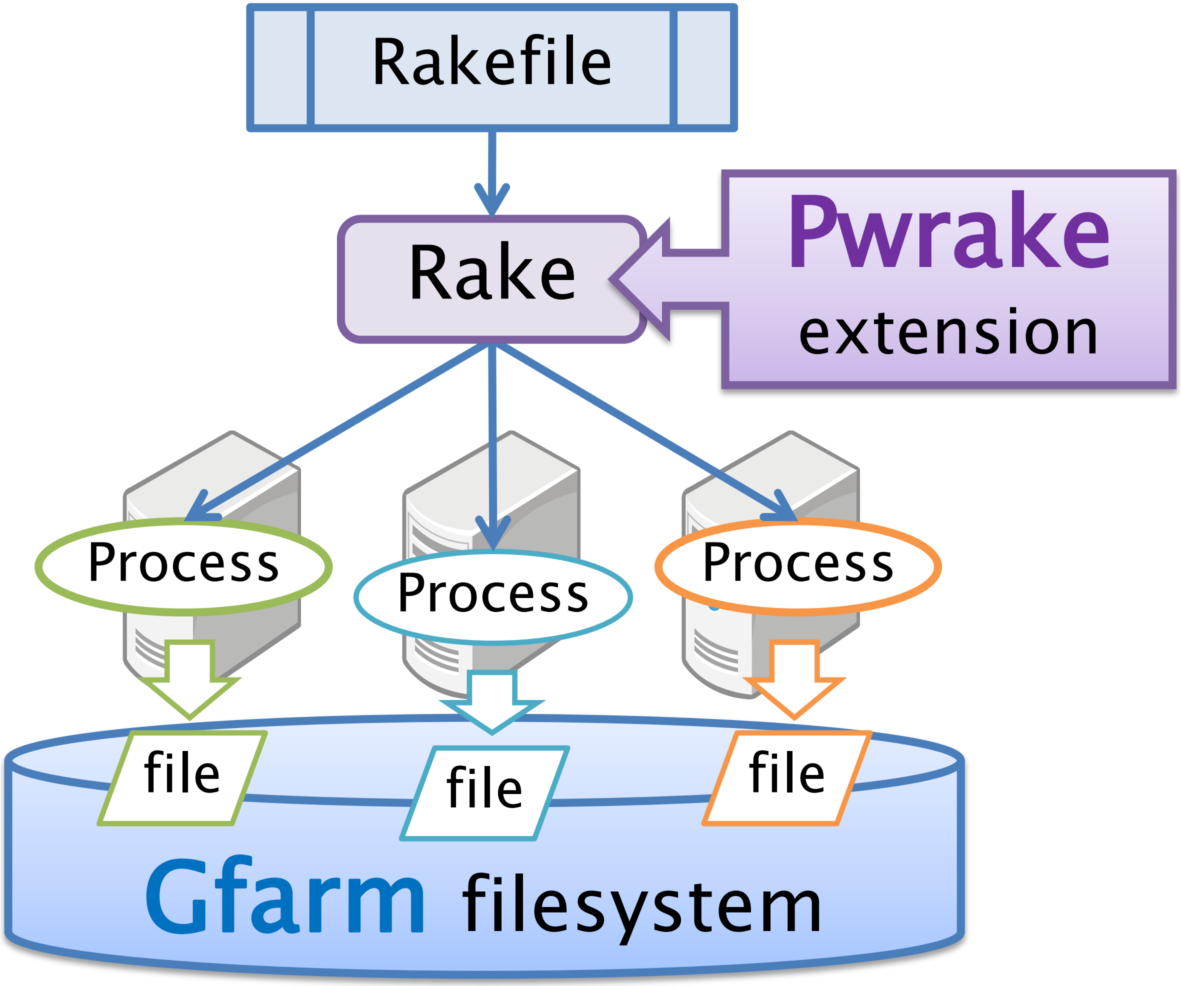
dRake does not have

- Remote process execution
- Dynamic task definition
  - dRake does not allows “invoke” method.
- Performance issue

Interim Summary

- Need Powerful Scientific Workflow tool
- Existing
  - Rake : Powerful for writing workflow
  - dRake : Parallel execution
- Missing
  - Remote Process Invocation
  - Scalability

Our Approach



Pwrake

Parallel Workflow extension for Rake

Pwrake

- Parallel + distributed
- Workflow
- extension for Rake

- repository:
  - http://github.com/masa16/pwrake

Pwrake features

- Same syntax as Rake.
- Parallelize **`task`, `file`**
  - no **multitask**
- Replace “**sh**” method
  - invoke process through SSH
- Scalability

Remote process call: SSH

- Why SSH
  - Secure
  - Probably SSH port is available
- SSH class for Pwrake
  - Original implementation
  - Performance issues

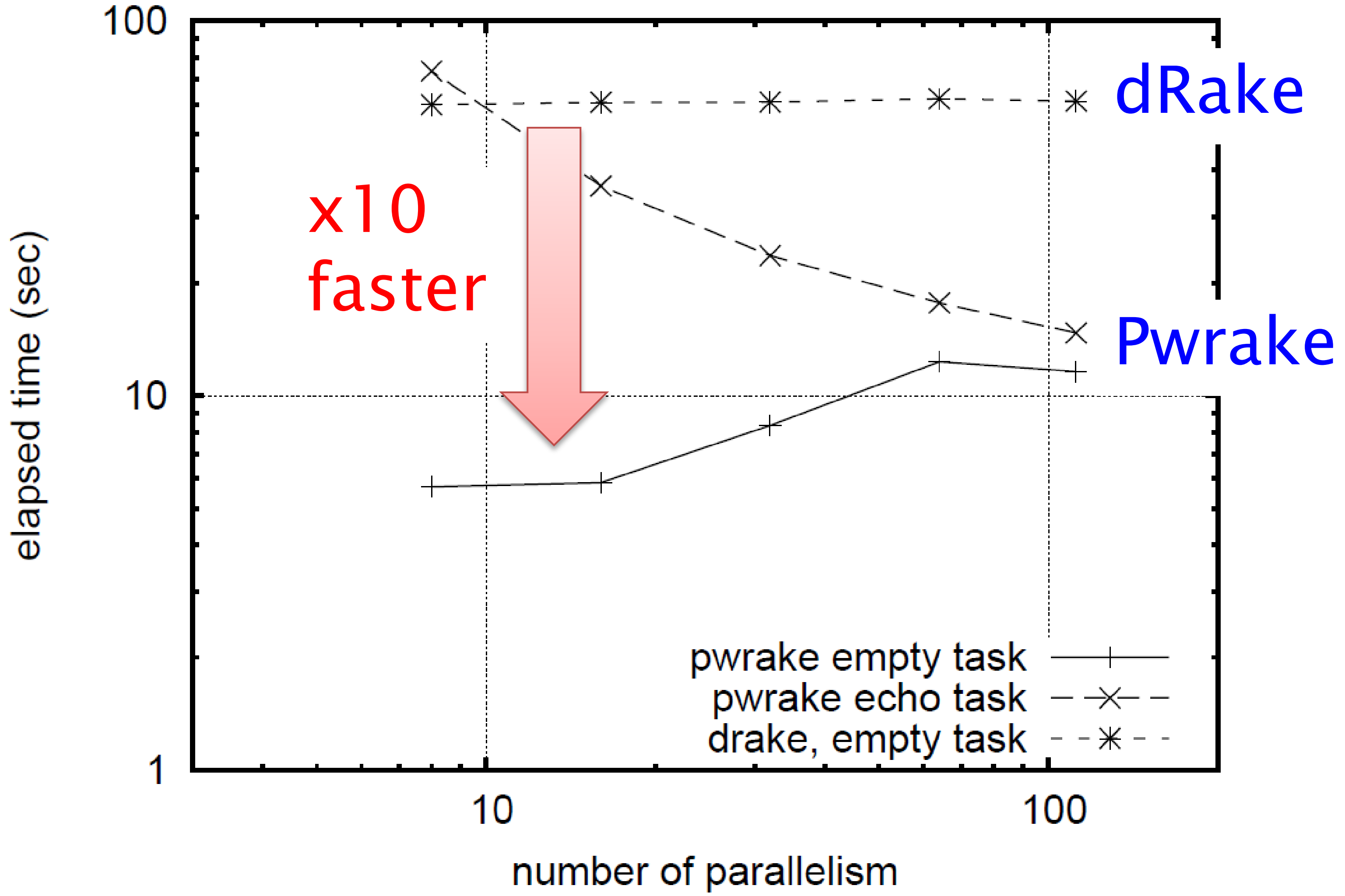
Pwrake Parallelism

- Worker thread in Ruby
- Ruby thread uses single-core
    - GVL
- **sh** process uses multi-core.

```
for x in LIST
  file x[1] => x[0] do |t|
    sh "your_program …"
  end
end
```

here uses multi-core

Pwrake performance : Empty tasks



dRake

Pwrake

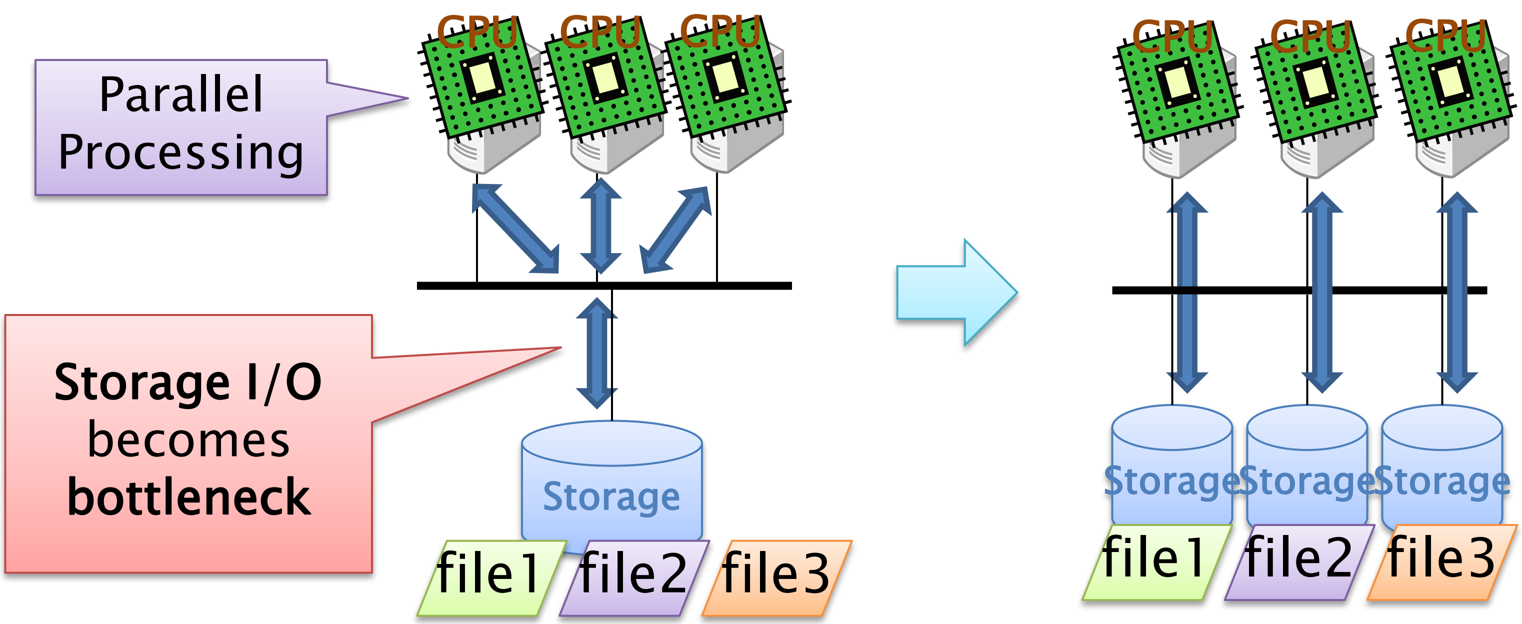
x10 faster

Remote File Access

Our approach:

- use **Distributed Filesystem**
  - file sharing
  - consistent file timestamp
  - I/O performance

File I/O is important for Data-intensive workflow



Network File System

Distributed File System

Parallel Processing

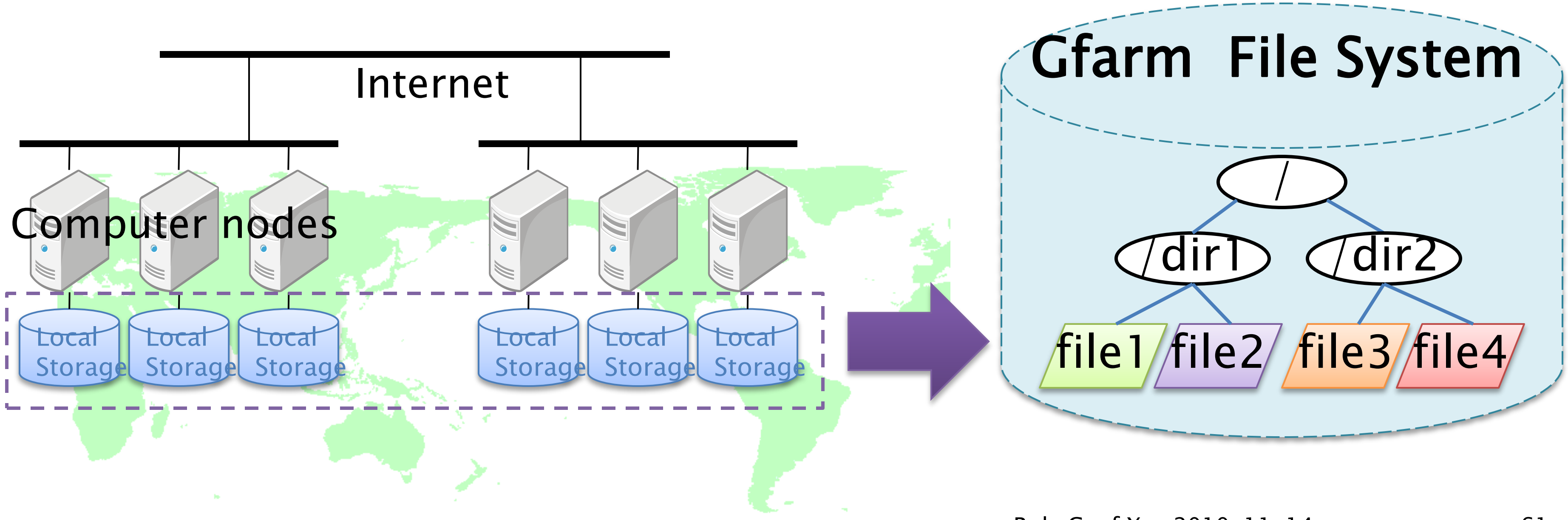
Storage I/O becomes **bottleneck**

Gfarm

Wide-area Distributed FileSystem

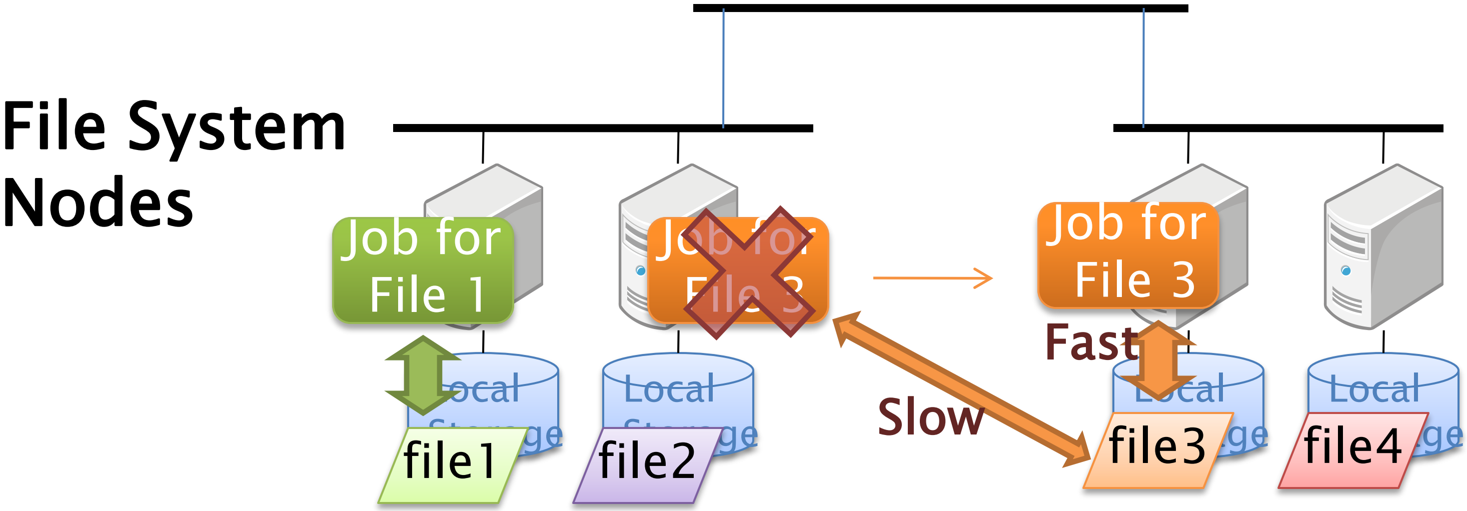
Gfarm

- Wide-area distributed file system
- Global namespace to federate storages
- Main developer : Prof. Osamu Tatebe
- Open source development
  - http://datafarm.apgrid.org/



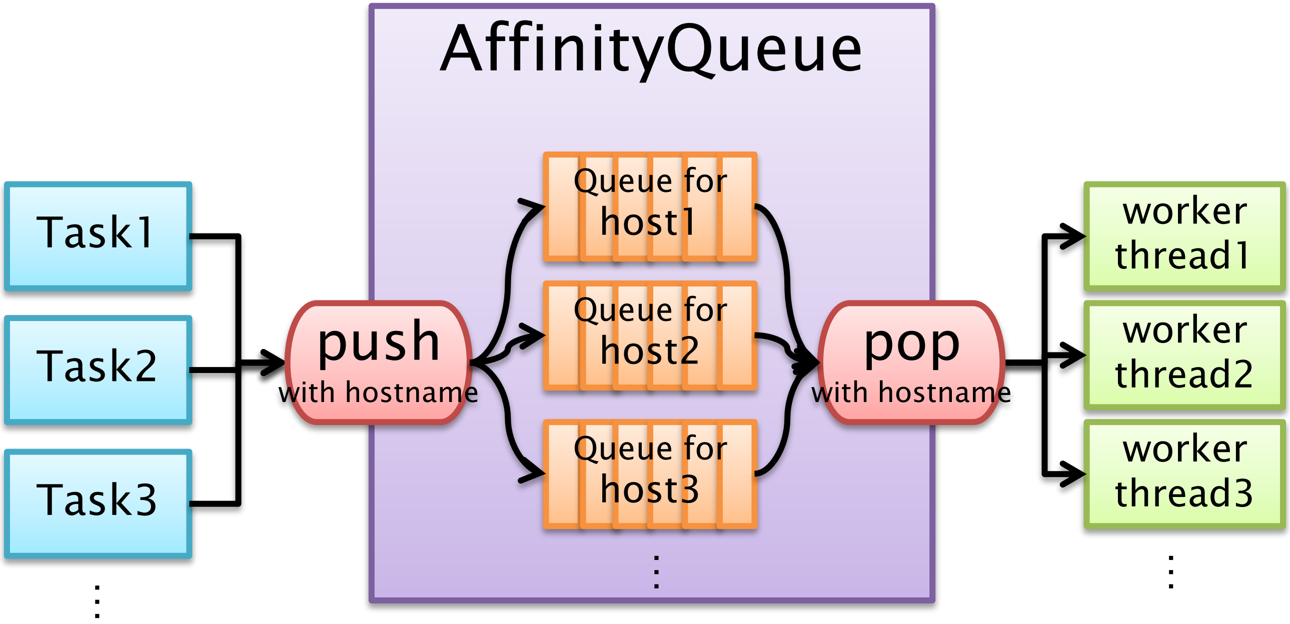
Gfarm unique feature

- use Local I/O for performance
- assign task based on File locality
- implement as a function of Pwrake

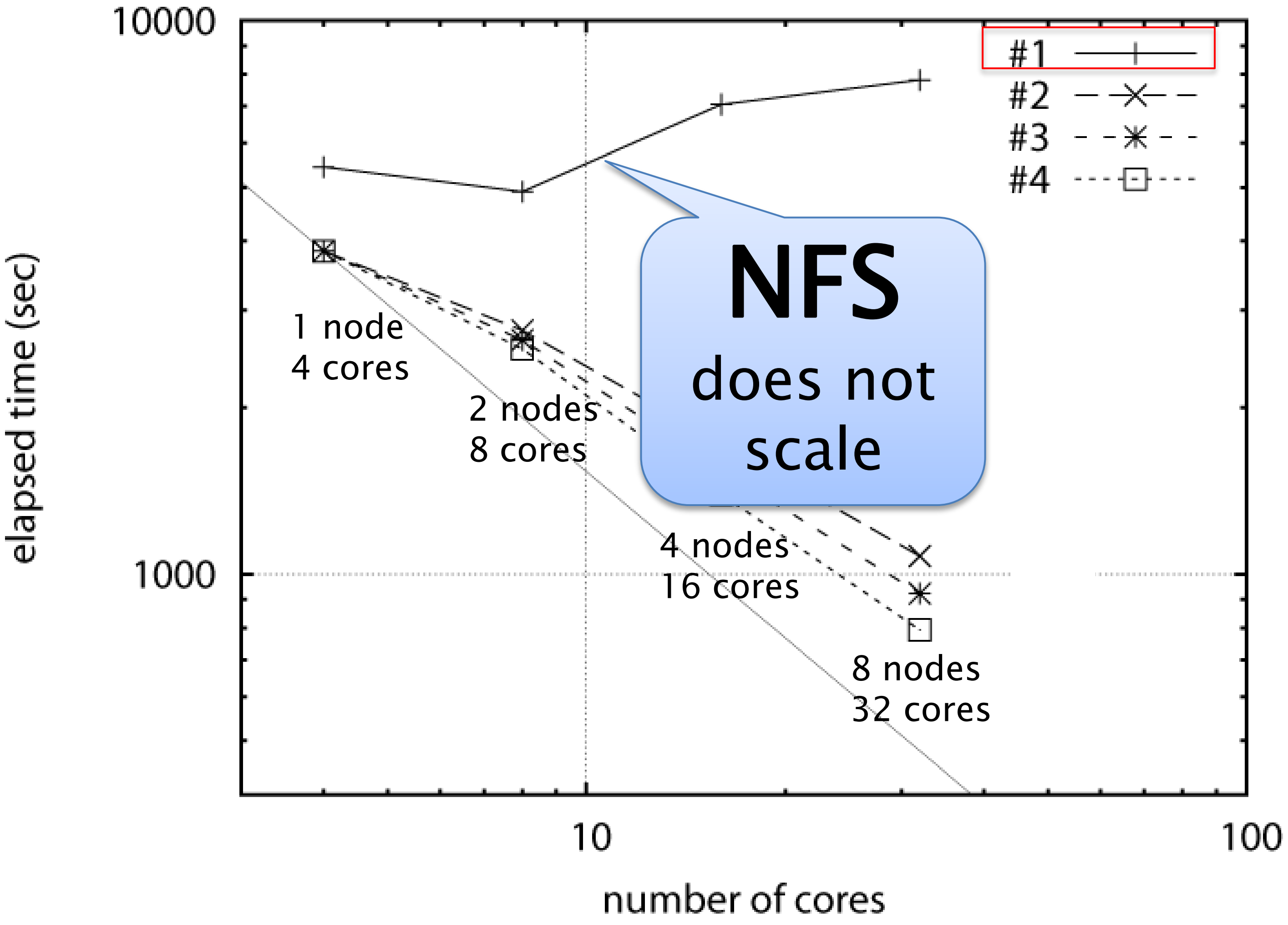


Locality algorithm for Pwrake

- Locality-aware task assignment for Gfarm



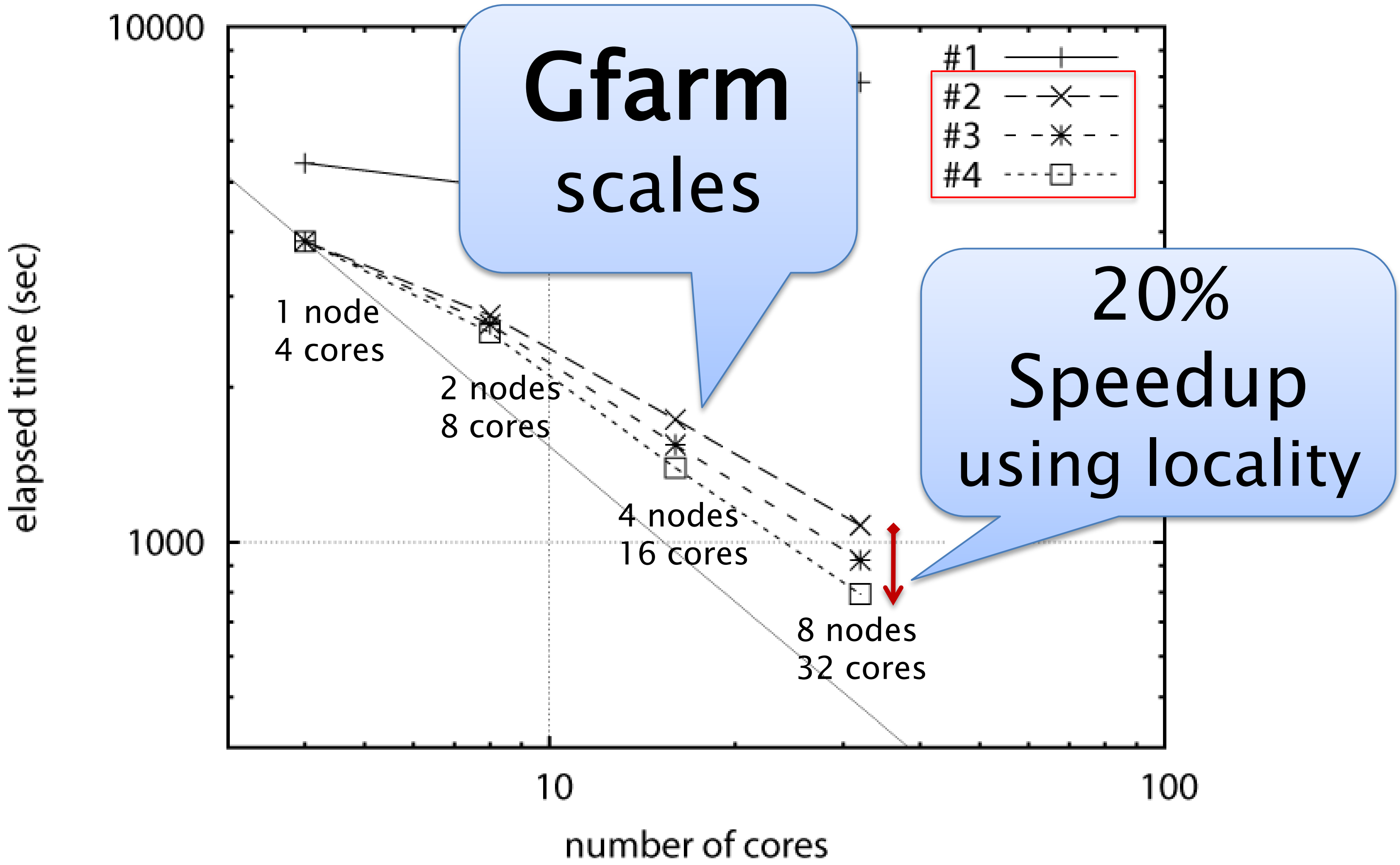
Performance of Montage workflow



NFS

does not scale

Performance of Montage workflow



Gfarm scales

20% Speedup using locality

Demo

- Montage workflow

Future Plan

- Geographically distributed wokrflow
- Fault tolerance

Conclusion

- Rake
  - is so powerful to be used for Scientific definition language.
- Pwrake
  - Parallel and Distributed Workflow extension for Rake
- Gfarm
  - for scalable I/O performance

Thank you for attension

- Pwrake site
  - https://github.com/masa16/pwrake
- Questions?