swift

Fast, Reliable, Loosely Coupled Parallel Computation

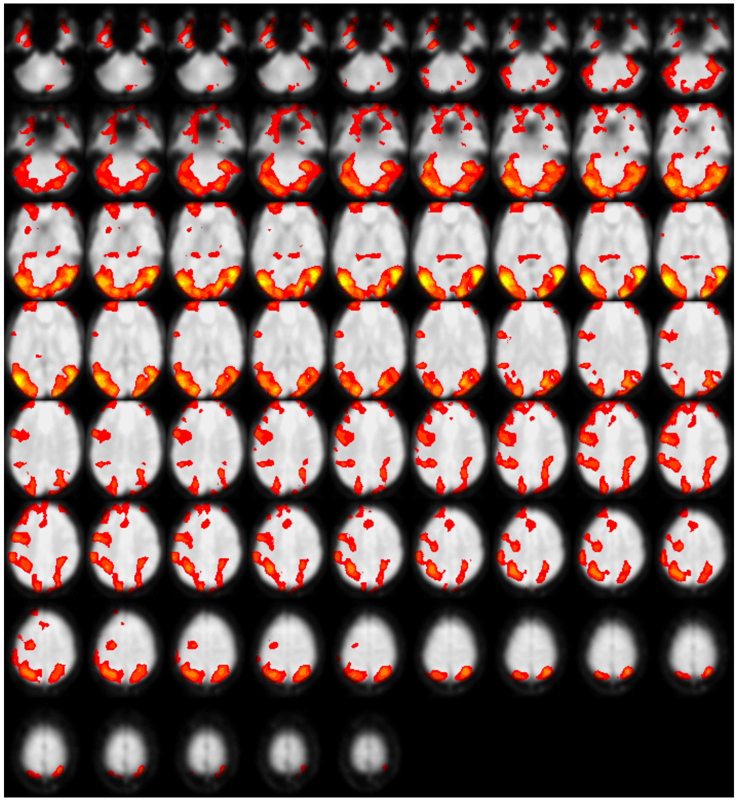
Ian Foster

Computation Institute
Argonne National Laboratory
University of Chicago

Joint work with **Yong Zhao, Ioan Raicu, Mike Wilde, Ben Clifford, Mihael Hatigan, Tibi Stef-Praun, Veronika Nefedova**

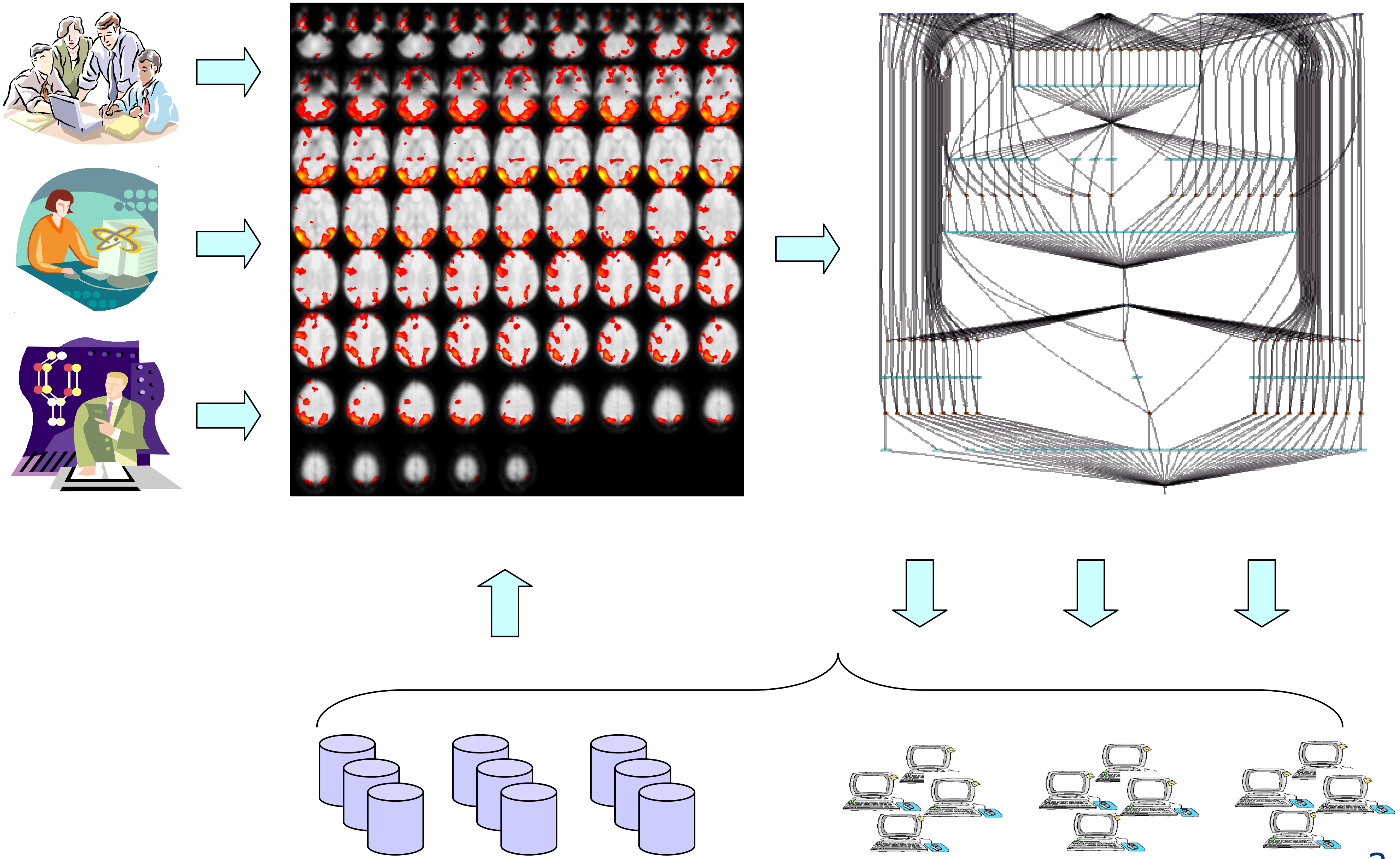
Case Study:
The Functional MRI (fMRI) Data Center

- Online repository of neuroimaging data
  - A typical study comprises 3 groups, 20 subjects/group, 5 runs/subject, 300 volumes/run
    → 90,000 volumes, 60 GB raw
    → 1.2 million files processed
  - 100s of such studies in total



- Many users analyze this data
  - Wide range of analyses
  - Testing → production
  - Ensembles: a set of data analyses by parameters, datasets

fMRI: A Broad Picture



Challenges

- Deluge of data: instrumentation, simulation
- Data analysis turns into data integration
- Community-wide collaboration
- Provenance: tracking, query, application
- Scalability: desktop to Grid
- Productivity: throughput, performance

Swift System

- Clean separation of logical/physical concerns
  - ◆ **XDTM** specification of logical data structures

+ Concise specification of parallel programs
  - ◆ **SwiftScript**, with iteration, etc.

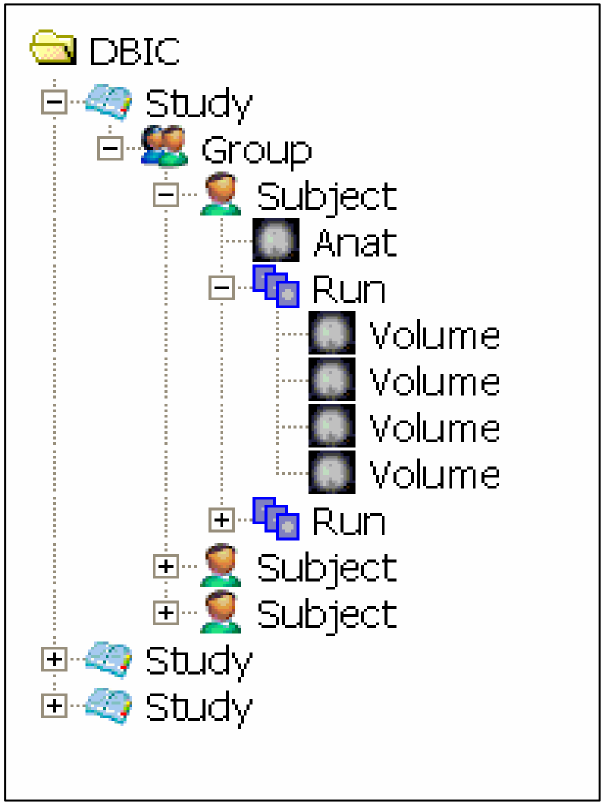
+ Efficient execution on distributed resources
  - ◆ **Karajan** threading, **Falkon** provisioning, **Globus** interfaces, pipelining, load balancing

+ Rigorous provenance tracking and query
  - ◆ Virtual data schema & automated recording

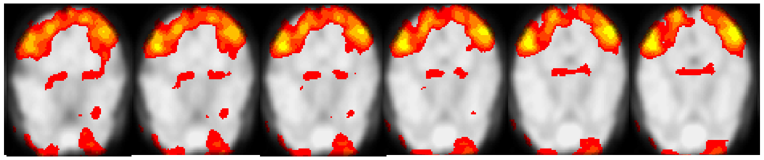
→ **Improved usability and productivity**
  - ◆ Demonstrated in numerous applications

The Messy Data Problem

- Scientific data is often logically structured
  - E.g., hierarchical structure
  - Common to map functions over dataset members
  - Nested map operations can scale to millions of objects



- DBIC
  - Study
    - Group
      - Subject
        - Anat
        - Run
          - Volume
          - Volume
          - Volume
          - Volume
        - Run
      - Subject
      - Subject
  - Study
  - Study



The Messy Data Problem

- But physically "messy"
- Heterogeneous storage format and access protocol
  - Logically identical dataset can be stored in textual File (e.g. CSV), spreadsheet, database, …
  - Data available from filesystem, DBMS, HTTP, WebDAV, ..
- Metadata encoded in directory and file names
- Hinders program development, composition, execution

```
./knottastic
total 58
drwxr-xr-x  4 yongzh users 2048 Nov 12 14:15 AA
drwxr-xr-x  4 yongzh users 2048 Nov 11 21:13 CH
drwxr-xr-x  4 yongzh users 2048 Nov 11 16:32 EC

./knottastic/AA:
total 4
drwxr-xr-x  5 yongzh users 2048 Nov  5 12:41 04nov06aa
drwxr-xr-x  4 yongzh users 2048 Dec  6 12:24 11nov06aa

. /knottastic//AA/04nov06aa:
total 54
drwxr-xr-x  2 yongzh users  2048 Nov  5 12:52 ANATOMY
drwxr-xr-x  2 yongzh users 49152 Dec  5 11:40 FUNCTIONAL

. /knottastic/AA/04nov06aa/ANATOMY:
total 58500
-rw-r--r--  1 yongzh users      348 Nov  5 12:29 coplanar.hdr
-rw-r--r--  1 yongzh users 16777216 Nov  5 12:29 coplanar.img

. /knottastic/AA/04nov06aa/FUNCTIONAL:
total 196739
-rw-r--r--  1 yongzh users    348 Nov  5 12:32 bold1_0001.hdr
-rw-r--r--  1 yongzh users  409600 Nov  5 12:32 bold1_0001.img
-rw-r--r--  1 yongzh users    348 Nov  5 12:32 bold1_0002.hdr
-rw-r--r--  1 yongzh users  409600 Nov  5 12:32 bold1_0002.img
-rw-r--r--  1 yongzh users    496 Nov 15 20:44 bold1_0002.mat
-rw-r--r--  1 yongzh users    348 Nov  5 12:32 bold1_0003.hdr
-rw-r--r--  1 yongzh users  409600 Nov  5 12:32 bold1_0003.img
```

XML Dataset Typing & Mapping (XDTM)

- Describe logical structure by **XML Schema**
  - Primitive scalar types: int, float, string, date, …
  - Complex types (structs and arrays)
- Use **mapping descriptors** for mappings
  - How dataset elements are mapped to physical representations
  - External parameters (e. g. location)
- Use **XPath** for dataset selection

XDTM: XML Dataset Typing and Mapping for Specifying Datasets [EGC05]

XDTM: Related Work

- Data format standardization
  - FITS, CDF, HDF-5, DICOM
- Data format description
  - DFDL [Beckerle,Westhead04] embeds annotations with XML Schema
  - PADS [Fisher,Gruber05], PADX [Fernandez,Fisher06], declarative specs of physical layout and semantic properties
- Logical object
  - ADO [Microsoft01], in memory relational model
  - SDO [Beatty,Brodsky03], logical data model for J2EE programming

XDTM: Implementation

- Virtual integration
  - Each data source treated as virtual XML source
  - Data structure defined as XML schema
  - Mapper responsible for accessing source and translating to/from XML representation
  - Bi-directional
- Common mapping interface
  - Data providers implement the interface
    - Responsible for data access details
  - Standard mapper implementations provided
    - String, file system, CSV, …

SwiftScript

- Typed parallel programm [SIGMOD05, Springer06]
  - ◆ XDTM as data model and type system
  - ◆ Typed dataset and procedure definitions
- Scripting language
  - ◆ Implicit data parallelism
  - ◆ Program composition from procedures
  - ◆ Control constructs (foreach, if, while, …)

Clean application logic
Type checking
Dataset selection, iteration
Discovery by types
Type conversion

A Notation & System for Expressing and Executing Cleanly Typed Workflows on Messy Scientific Data [SIGMOD05]

SwiftScript: Related Work

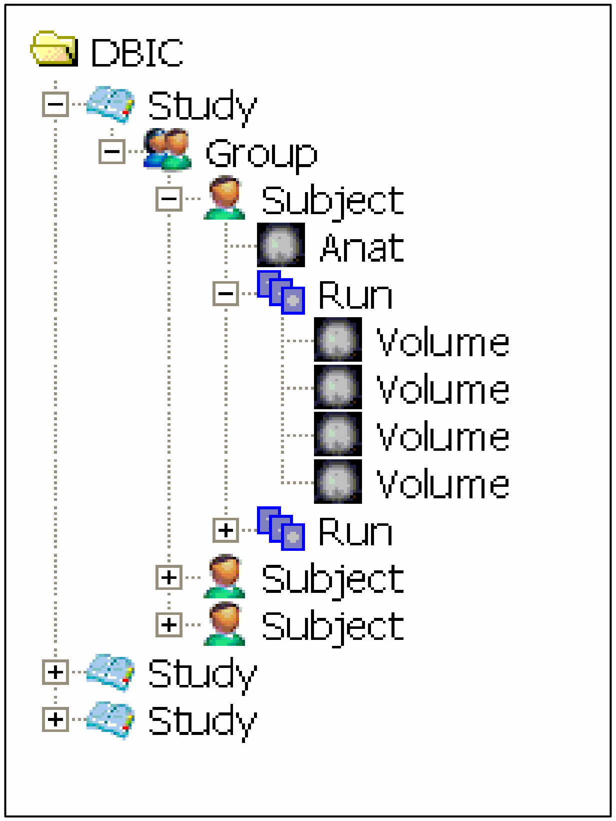
- Coordination language
  - Linda[Ahuja,Carriero86], Strand[Foster,Taylor90], PCN[Foster92]
  - Durra[Barbacci,Wing86], MANIFOLD[Papadopoulos98]
  - Components programmed in specific language (C, FORTRAN) and linked with system
- “Workflow” languages and systems
  - Taverna[Oinn,Addis04], Kepler[Ludäscher,Altintas05], Triana [Churches,Gombas05], Vistrail[Callahan,Freire06], DAGMan, Star-P
  - XPDL[WfMC02], BPEL[Andrews,Curbera03], and BPML[BPML02], YAWL[van de Aalst,Hofstede05], Windows Workflow Foundation [Microsoft05]

Related Work

	SwiftScript	BPEL	XPDL	MW Wflow	DAGMan	Tavena	Triana	Kepler	Vistrail	Star-P
Scales to Grids	++	-	-	-	++	-	-	-	-	+
Typing	++	++	++	++	-	-	-	+	-	+
Iteration	++	-/+	-	+	-	-	-	+	-	+
Scripting	++	-	-	+	+	+	-	-	+	++
Dataset Mapping	+	-	-	-	-	-	-	-	-	-
Service Interop	+	-	+	-	-	-	-	+	-	-
Subflow/comp.	+	-	+	+	-	-	+	+	-	+
Provenance	+	-	-	+	-	+	-	+	+	-
Open source	+	+	+	-	+	+	+	+	+	-

"A 4x200 flow leads to a 5 MB BPEL file … chemists were not able to write in BPEL" [Emmerich,Buchart06]

fMRI Type Definitions in SwiftScript



Simplified version of fMRI AIRSN Program (Spatial Normalization)

```
type Study {
        Group g[ ];
}

type Group {
        Subject s[ ];
}

type Subject {
        Volume anat;
        Run run[ ];
}

type Run {
        Volume v[ ];
}

type Volume {
        Image img;
        Header hdr;
}
```

```
type Image {};

type Header {};

type Warp {};

type Air {};

type AirVec {
        Air a[ ];
}

type NormAnat {
        Volume anat;
        Warp aWarp;
        Volume nHires;
}
```

Type Definitions in XML Schema

```
<xs:schema targetNamespace="http://www.fmri.org/schema/airsn.xsd"
                  xmlns="http://www.fmri.org/schema/airsn.xsd"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema">
              <xs:simpleType name="Image">
                     <xs:restriction base="xs:string"/>
              </xs:simpleType>
              <xs:simpleType name="Header">
                     <xs:restriction base="xs:string"/>
              </xs:simpleType>
              <xs:complexType name="Volume">
                     <xs:sequence>
                            <xs:element name="img" type="Image"/>
                            <xs:element name="hdr" type="Header"/>
                     </xs:sequence>
              </xs:complexType>
              <xs:complexType name="Run">
                     <xs:sequence minOccurs="0” maxOccurs="unbounded">
                            <xs:element name="v" type="Volume"/>
                      </xs:sequence>
              </xs:complexType>
<xs:schema>
```

AIRSN Program Definition

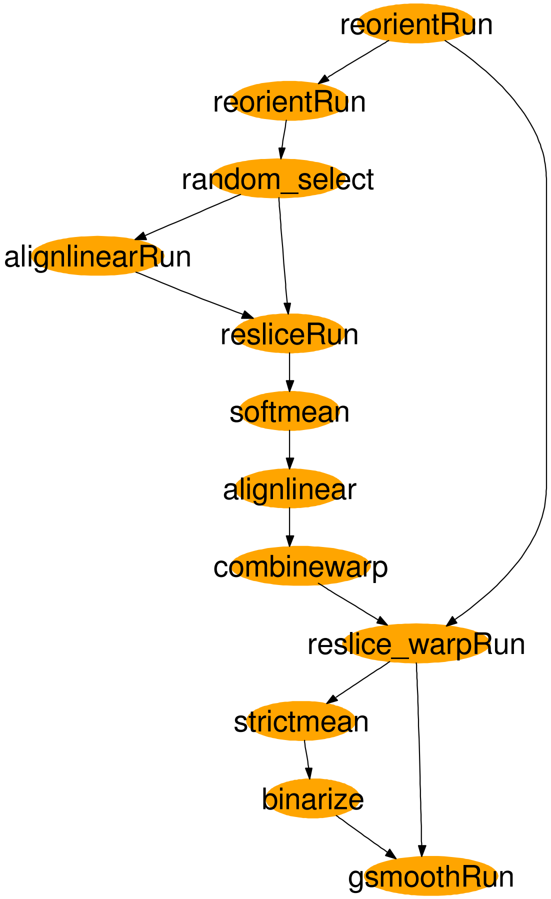
```
(Run or) reorientRun (Run ir,
                          string direction) {
      foreach Volume iv, i in ir.v {
            or.v[i] = reorient(iv, direction);
      }
  }
```

```
(Run snr) functional ( Run r, NormAnat a,
                           Air shrink ) {
    Run yroRun = reorientRun( r , "y" );
    Run roRun = reorientRun( yroRun , "x" );
    Volume std = roRun[0];
    Run rndr = random_select( roRun, 0.1 );
    AirVector rndAirVec = align_linearRun( rndr, std, 12, 1000, 1000, "81 3 3" );
    Run reslicedRndr = resliceRun( rndr, rndAirVec, "o", "k" );
    Volume meanRand = softmean( reslicedRndr, "y", "null" );
    Air mnQAAir = alignlinear( a.nHires, meanRand, 6, 1000, 4, "81 3 3" );
    Warp boldNormWarp = combinewarp( shrink, a.aWarp, mnQAAir );
    Run nr = reslice_warp_run( boldNormWarp, roRun );
    Volume meanAll = strictmean( nr, "y", "null" )
    Volume boldMask = binarize( meanAll, "y" );
    snr = gsmoothRun( nr, boldMask, "6 6 6" );
}
```

Expressiveness

Lines of code with different encodings

Appln	Script	Generator	Swift Script
ATLAS1	49	72	**6**
ATLAS2	97	135	**10**
FILM1	63	134	**17**
FEAT	84	191	**13**
AIRSN	215	~400	**34**

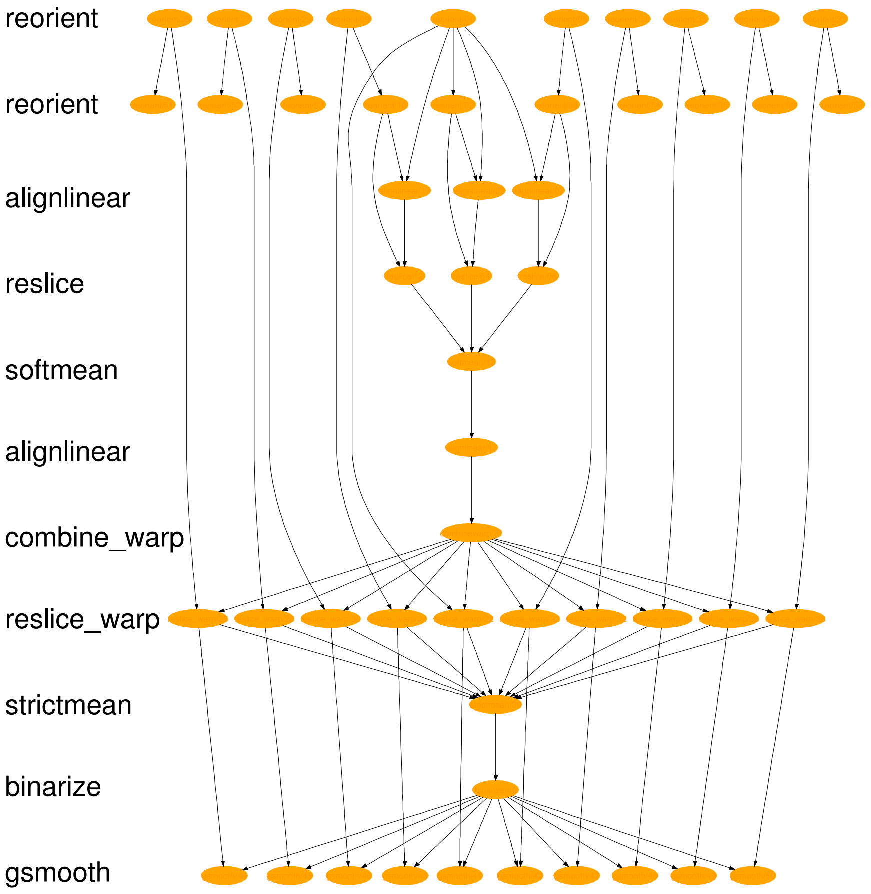


Collaboration with James Dobson, Dartmouth [SIGMOD05]

Expressiveness

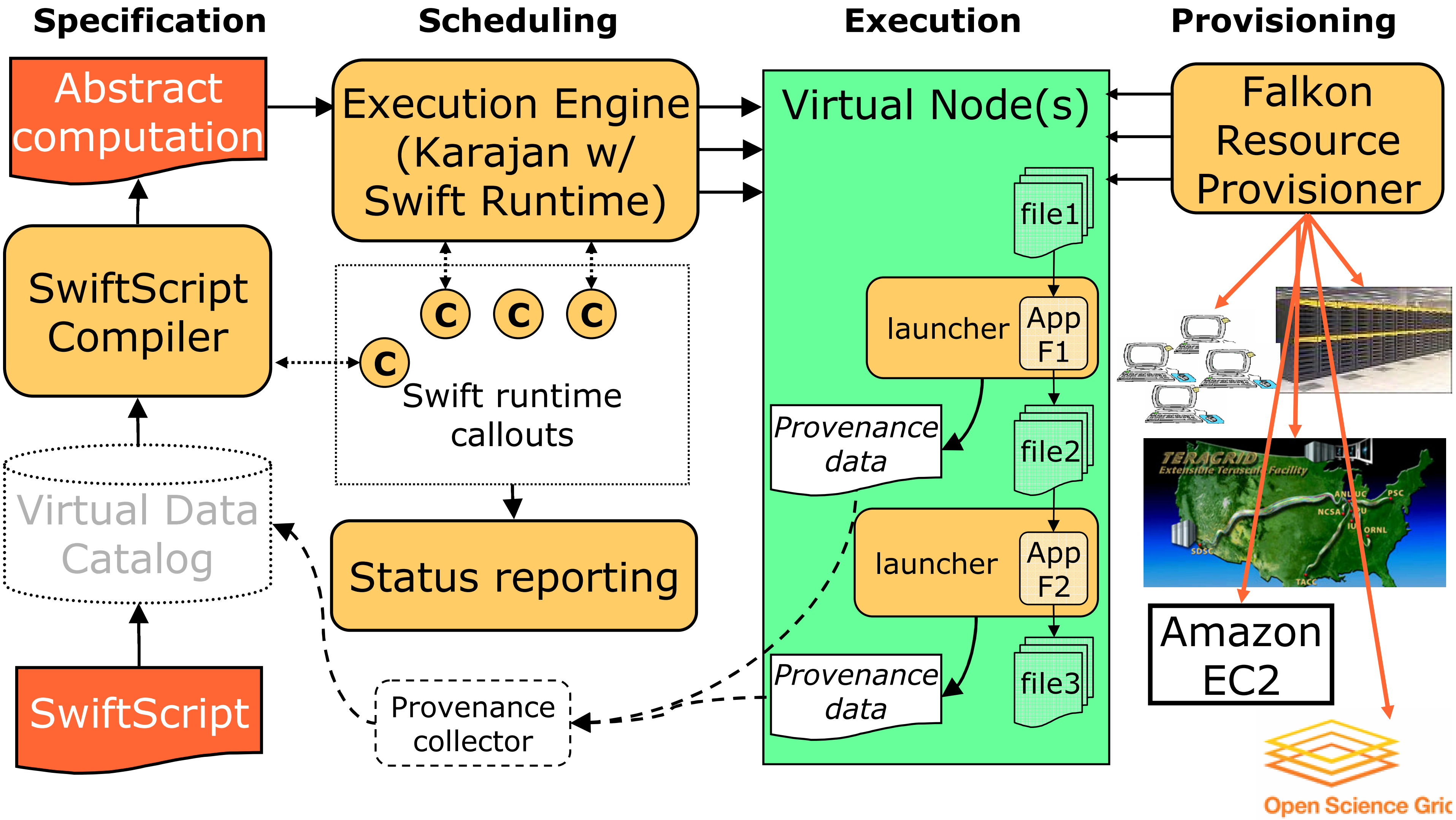
Lines of code with different encodings

Appln	Script	Generator	Swift Script
ATLAS1	49	72	**6**
ATLAS2	97	135	**10**
FILM1	63	134	**17**
FEAT	84	191	**13**
AIRSN	215	~400	**34**



Collaboration with James Dobson, Dartmouth [SIGMOD05]

Dynamic Provisioning: Swift Architecture



Specification

Scheduling

Execution

Provisioning

Abstract computation

SwiftScript Compiler

Virtual Data Catalog

SwiftScript

Execution Engine (Karajan w/ Swift Runtime)

C C C C

Swift runtime callouts

Status reporting

Provenance collector

Virtual Node(s)

file1

launcher App F1

file2

launcher App F2

file3

Provenance data

Provenance data

Falkon Resource Provisioner

Amazon EC2

Yong Zhao, Mihael Hatigan, Ioan Raicu, Mike Wilde, Ben Clifford

Swift Runtime System

- Runtime system for SwiftScript [SSDBM02,CIDR03,Springer06]
  - Populate, update, query virtual data products
  - Schedule, monitor, execute resulting computation on distributed Grid resources
  - Annotate virtual data products with customized metadata
  - Trace provenance of virtual data products
- Grid scheduling and optimization
  - Lightweight execution engine: Karajan
  - Dynamic resource provisioning
  - Site selection, data movement, caching
  - Pipelining, clustering, load balancing
  - Fault tolerance, exception handling

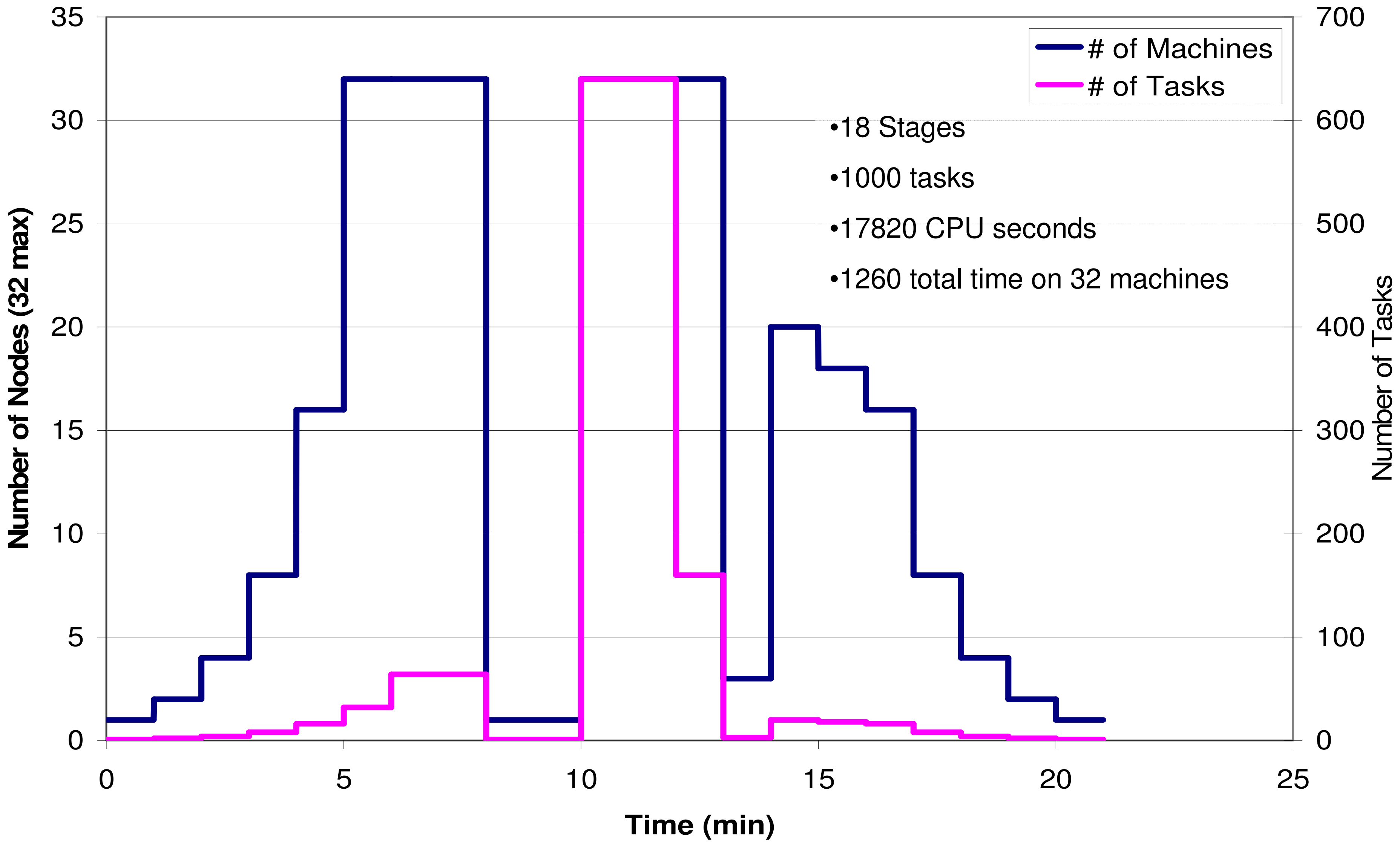
A Virtual Data System for Representing, Querying & Automating Data Derivation [SSDBM02]

Swift uses Karajan Workflow Engine

- Fast, scalable threading model
- Suitable constructs for control flow
- Flexible task dependency model
  - "Futures" enable pipelining
- Flexible provider model allows for use of different run time environments
  - Job execution and data transfer
  - Flow controlled to avoid resource overload
- Workflow client runs from a Java container

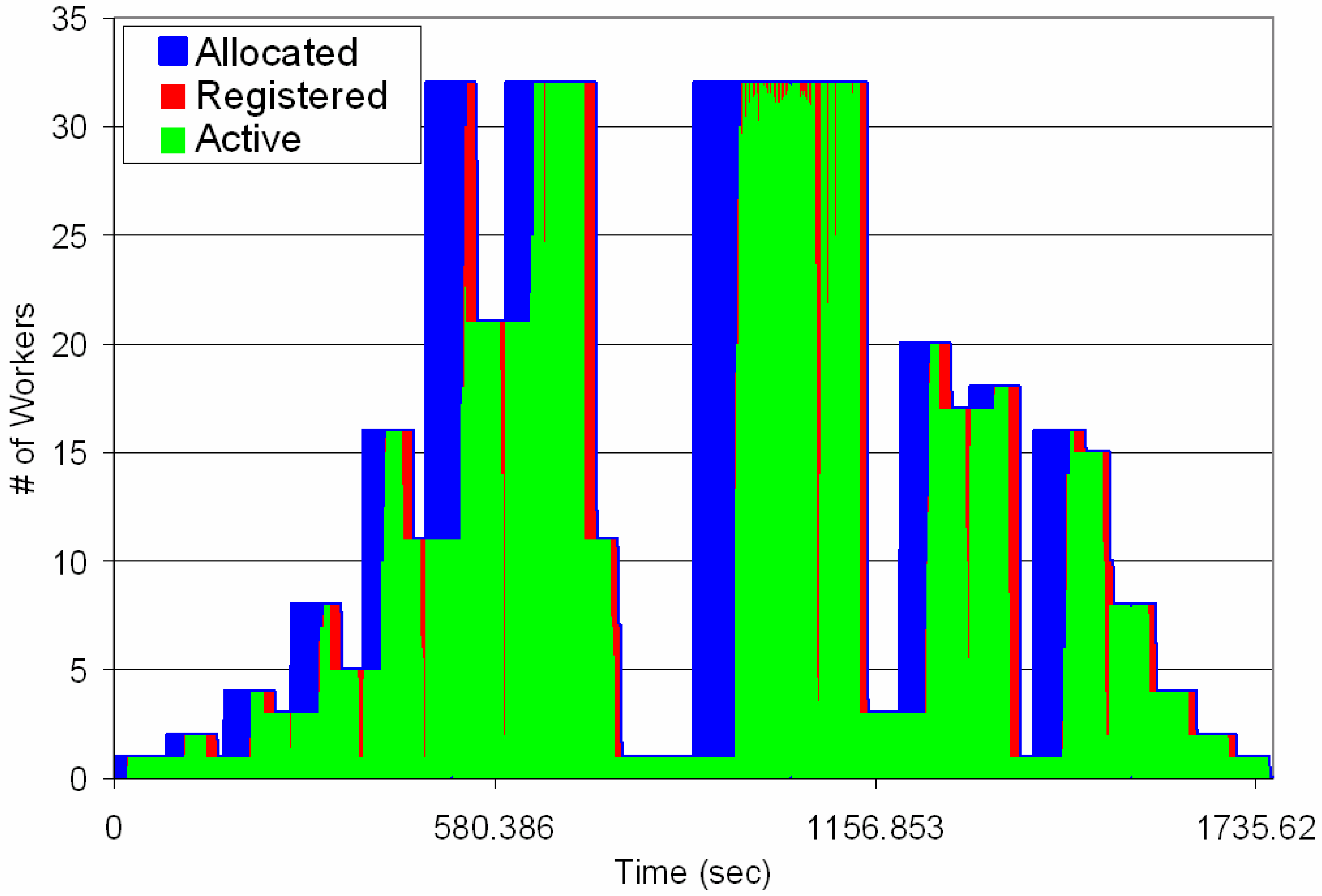
Java CoG Workflow, Gregor von Laszewski et al., *Workflows for Science*, 2007

Synthetic Benchmark

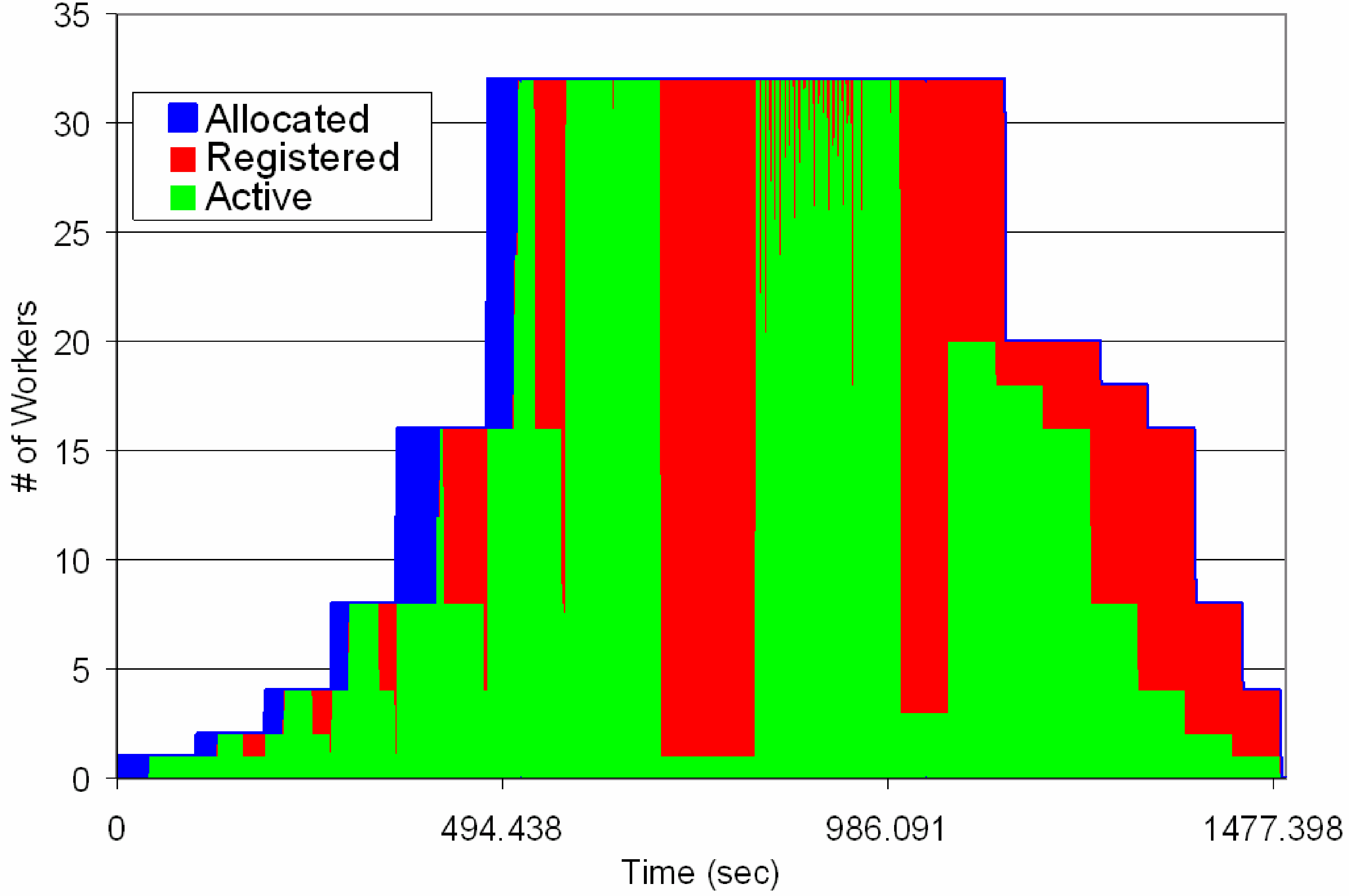


- 18 Stages
- 1000 tasks
- 17820 CPU seconds
- 1260 total time on 32 machines

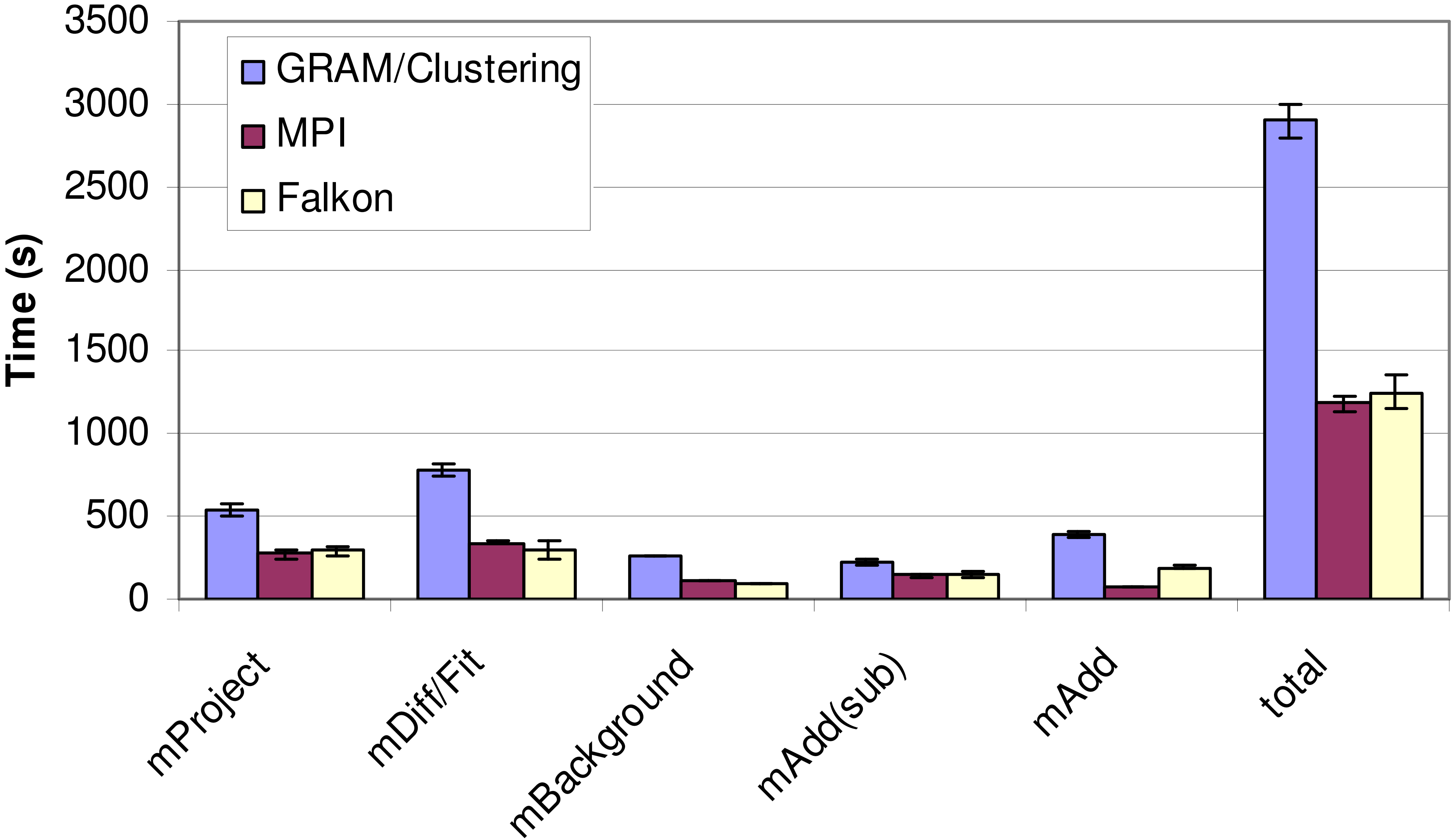
Release after 15 Seconds Idle



Release after 180 Seconds Idle

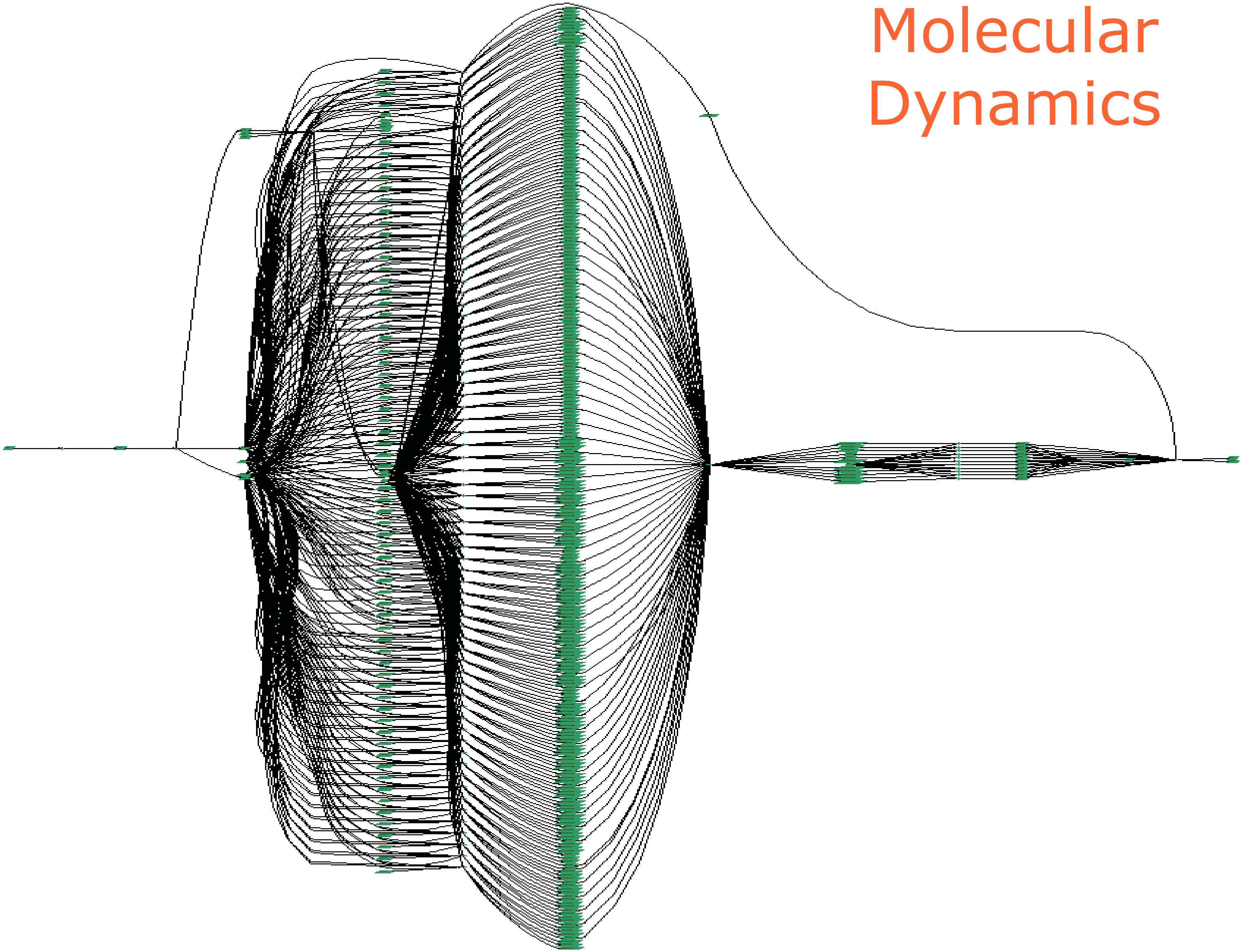


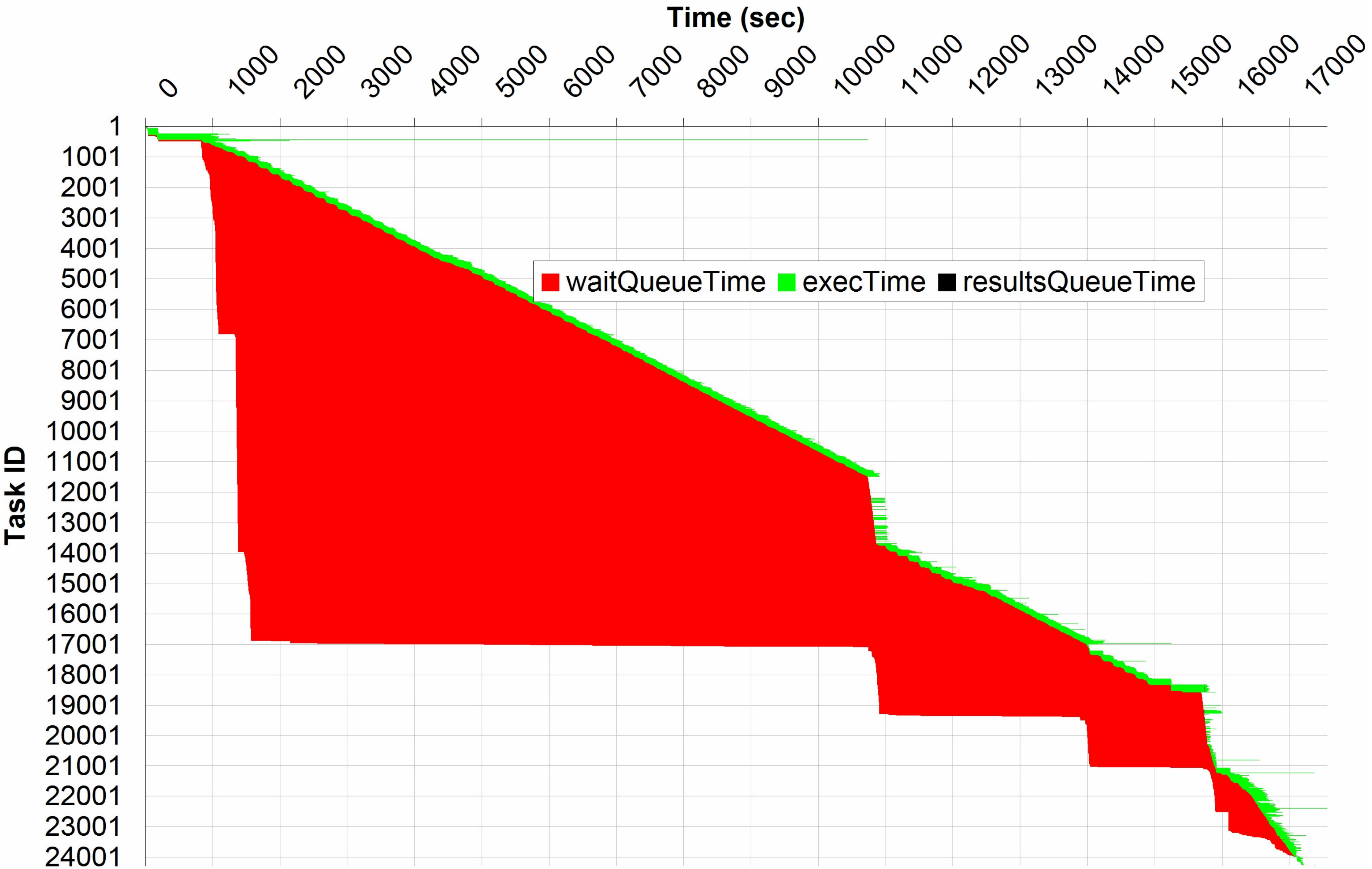
Montage

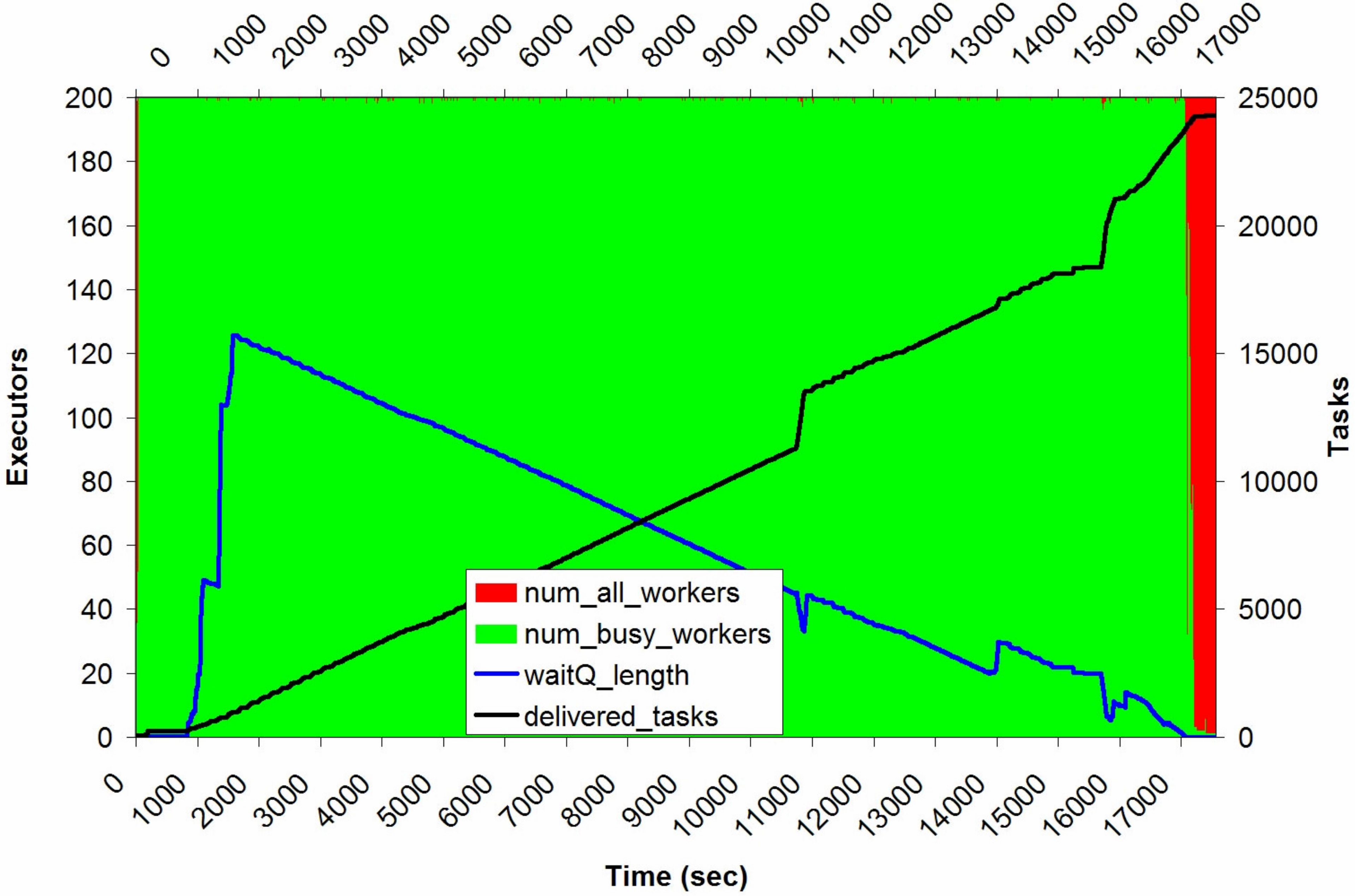


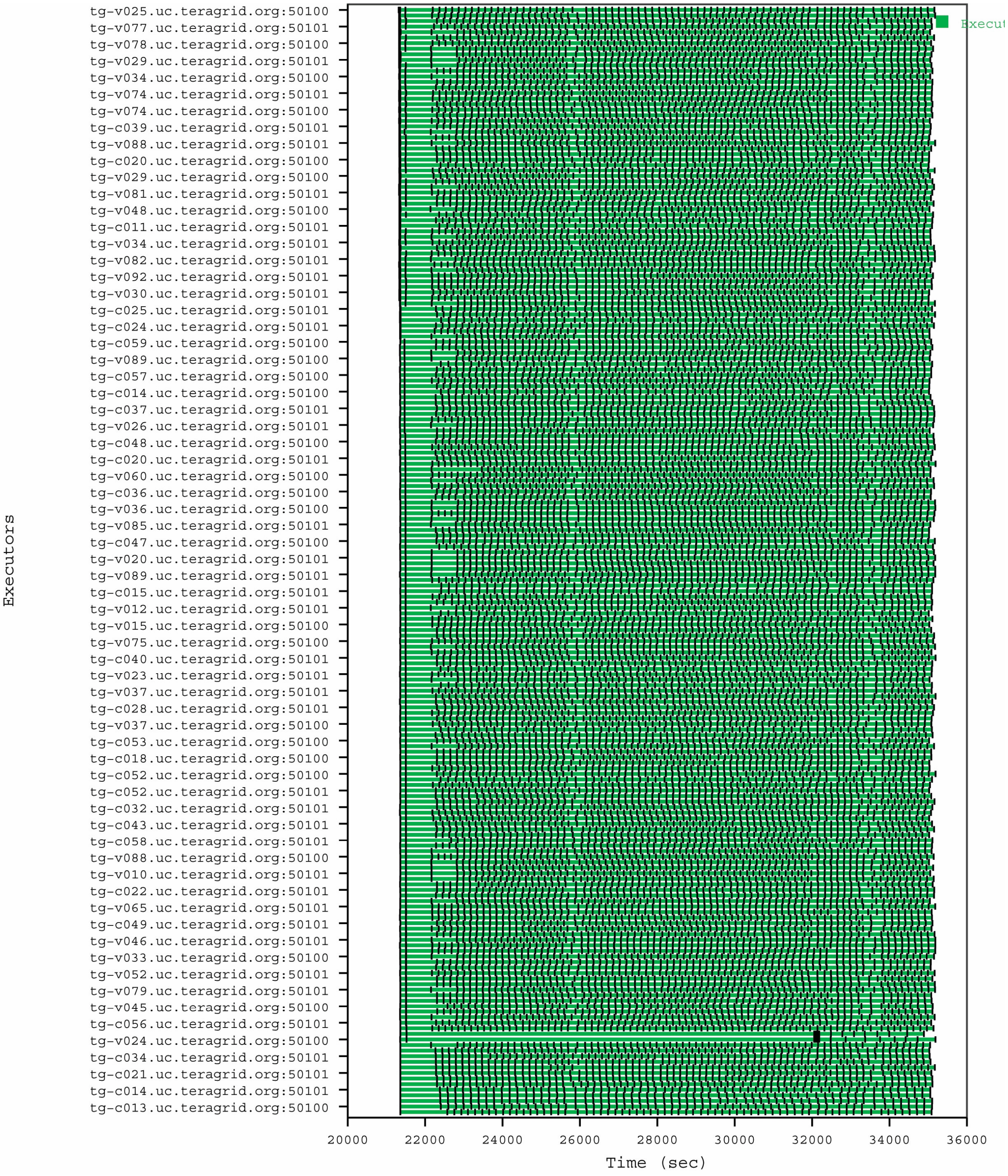
Yong Zhao and Ioan Raicu, U.Chicago

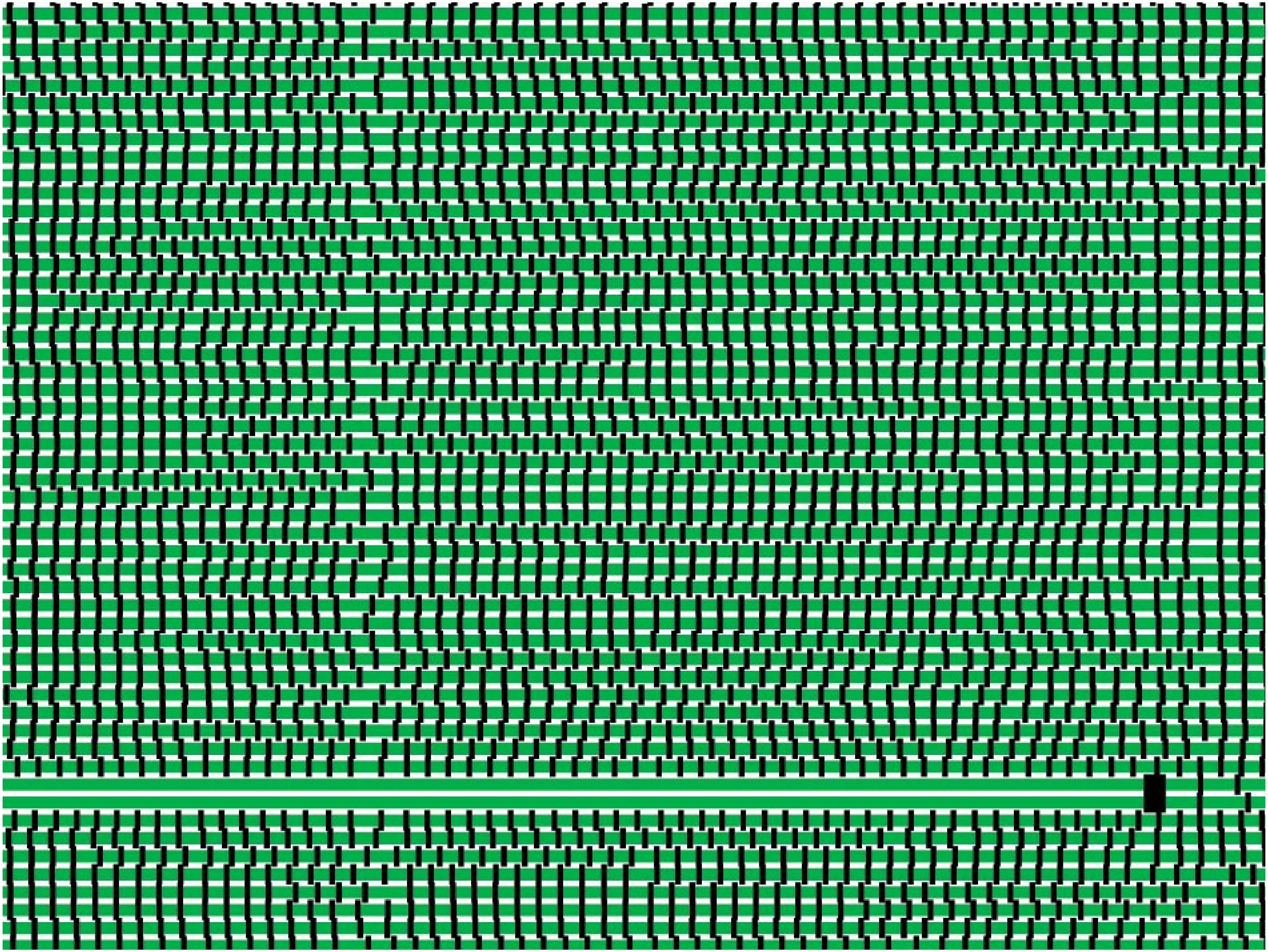
Molecular Dynamics



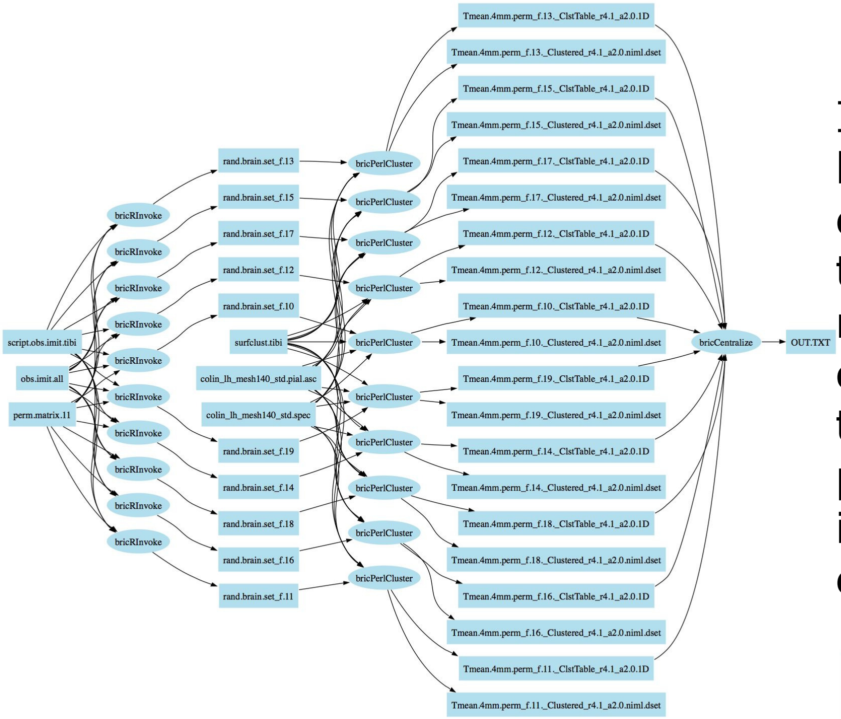

Time (sec)
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
12000
13000
14000
15000
16000
17000
Task ID
1
1001
2001
3001
4001
5001
6001
7001
8001
9001
10001
11001
12001
13001
14001
15001
16001
17001
18001
19001
20001
21001
22001
23001
24001
🟥 waitQueueTime
🟩 execTime
■ resultsQueueTime


0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
12000
13000
14000
15000
16000
17000
Executors
200
180
160
140
120
100
80
60
40
20
0
Tasks
25000
20000
15000
10000
5000
0
🟥 num_all_workers
🟩 num_busy_workers
waitQ_length
delivered_tasks
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
12000
13000
14000
15000
16000
17000
Time (sec)

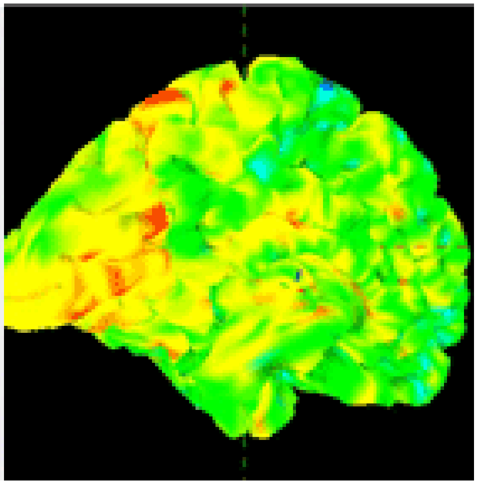

tg-v025.uc.teragrid.org:50100
tg-v077.uc.teragrid.org:50101
tg-v078.uc.teragrid.org:50100
tg-v029.uc.teragrid.org:50101
tg-v034.uc.teragrid.org:50100
tg-v074.uc.teragrid.org:50101
tg-v074.uc.teragrid.org:50100
tg-c039.uc.teragrid.org:50101
tg-v088.uc.teragrid.org:50101
tg-c020.uc.teragrid.org:50100
tg-v029.uc.teragrid.org:50100
tg-v081.uc.teragrid.org:50101
tg-v048.uc.teragrid.org:50100
tg-c011.uc.teragrid.org:50101
tg-v034.uc.teragrid.org:50101
tg-v082.uc.teragrid.org:50101
tg-v092.uc.teragrid.org:50101
tg-v030.uc.teragrid.org:50101
tg-c025.uc.teragrid.org:50101
tg-c024.uc.teragrid.org:50101
tg-c059.uc.teragrid.org:50100
tg-v089.uc.teragrid.org:50100
tg-c057.uc.teragrid.org:50100
tg-c014.uc.teragrid.org:50100
tg-c037.uc.teragrid.org:50101
tg-v026.uc.teragrid.org:50101
tg-c048.uc.teragrid.org:50100
tg-c020.uc.teragrid.org:50101
tg-v060.uc.teragrid.org:50100
tg-c036.uc.teragrid.org:50100
tg-v036.uc.teragrid.org:50100
tg-v085.uc.teragrid.org:50101
tg-c047.uc.teragrid.org:50100
tg-v020.uc.teragrid.org:50101
tg-v089.uc.teragrid.org:50101
tg-c015.uc.teragrid.org:50101
tg-v012.uc.teragrid.org:50101
tg-v015.uc.teragrid.org:50100
tg-v075.uc.teragrid.org:50100
tg-c040.uc.teragrid.org:50101
tg-v023.uc.teragrid.org:50101
tg-v037.uc.teragrid.org:50101
tg-c028.uc.teragrid.org:50101
tg-v037.uc.teragrid.org:50100
tg-c053.uc.teragrid.org:50100
tg-c018.uc.teragrid.org:50100
tg-c052.uc.teragrid.org:50100
tg-c052.uc.teragrid.org:50101
tg-c032.uc.teragrid.org:50101
tg-c043.uc.teragrid.org:50101
tg-c058.uc.teragrid.org:50101
tg-v088.uc.teragrid.org:50100
tg-v010.uc.teragrid.org:50101
tg-c022.uc.teragrid.org:50101
tg-v065.uc.teragrid.org:50101
tg-c049.uc.teragrid.org:50101
tg-v046.uc.teragrid.org:50101
tg-v033.uc.teragrid.org:50100
tg-v052.uc.teragrid.org:50101
tg-v079.uc.teragrid.org:50101
tg-v045.uc.teragrid.org:50100
tg-c056.uc.teragrid.org:50101
tg-v024.uc.teragrid.org:50100
tg-c034.uc.teragrid.org:50101
tg-c021.uc.teragrid.org:50101
tg-c014.uc.teragrid.org:50101
tg-c013.uc.teragrid.org:50100
Executors
20000
22000
24000
26000
28000
30000
32000
34000
36000
Time (sec)
🟩 Execut



Application Example:
ACTIVAL: Neural Activation Validation


Tmean.4mm.perm_f.13._ClstTable_r4.1_a2.0.1D
Tmean.4mm.perm_f.13._Clustered_r4.1_a2.0.niml.dset
Tmean.4mm.perm_f.15._ClstTable_r4.1_a2.0.1D
Tmean.4mm.perm_f.15._Clustered_r4.1_a2.0.niml.dset
rand.brain.set_f.13
bricPerlCluster
Tmean.4mm.perm_f.17._ClstTable_r4.1_a2.0.1D
rand.brain.set_f.15
bricPerlCluster
Tmean.4mm.perm_f.17._Clustered_r4.1_a2.0.niml.dset
bricRInvoke
rand.brain.set_f.17
Tmean.4mm.perm_f.12._ClstTable_r4.1_a2.0.1D
bricRInvoke
bricPerlCluster
rand.brain.set_f.12
Tmean.4mm.perm_f.12._Clustered_r4.1_a2.0.niml.dset
bricRInvoke
bricPerlCluster
rand.brain.set_f.10
Tmean.4mm.perm_f.10._ClstTable_r4.1_a2.0.1D
bricRInvoke
script.obs.imit.tibi
surfclust.tibi
bricPerlCluster
Tmean.4mm.perm_f.10._Clustered_r4.1_a2.0.niml.dset
bricCentralize
OUT.TXT
bricRInvoke
obs.imit.all
colin_lh_mesh140_std.pial.asc
Tmean.4mm.perm_f.19._ClstTable_r4.1_a2.0.1D
bricRInvoke
bricPerlCluster
perm.matrix.11
colin_lh_mesh140_std.spec
Tmean.4mm.perm_f.19._Clustered_r4.1_a2.0.niml.dset
bricRInvoke
bricPerlCluster
rand.brain.set_f.19
Tmean.4mm.perm_f.14._ClstTable_r4.1_a2.0.1D
bricRInvoke
rand.brain.set_f.14
bricPerlCluster
Tmean.4mm.perm_f.14._Clustered_r4.1_a2.0.niml.dset
bricRInvoke
rand.brain.set_f.18
Tmean.4mm.perm_f.18._ClstTable_r4.1_a2.0.1D
bricRInvoke
bricPerlCluster
Tmean.4mm.perm_f.18._Clustered_r4.1_a2.0.niml.dset
rand.brain.set_f.16
bricPerlCluster
rand.brain.set_f.11
Tmean.4mm.perm_f.16._ClstTable_r4.1_a2.0.1D
Tmean.4mm.perm_f.16._Clustered_r4.1_a2.0.niml.dset
Tmean.4mm.perm_f.11._ClstTable_r4.1_a2.0.1D
Tmean.4mm.perm_f.11._Clustered_r4.1_a2.0.niml.dset

Identifies clusters of neural activity not likely to be active by random chance: switch labels of the conditions for one or more participants; calculate the delta values in each voxel, re-calculate the reliability of delta in each voxel, and evaluate clusters found. If the clusters in data are greater than the majority of the clusters found in the permutations, then the null hypothesis is refuted indicating that clusters of activity found in our experiment are not likely to be found by chance.



Work by S. Small,
U. Hasson, UChicago.

SwiftScript Program ACTIVAL – Datatypes & Utilities

```
type script {}
type brainMeasurements{}
type precomputedPermutations{}
type brainClusterTable {}
type brainDatasets{ brainDataset b[]; }
type brainClusters{ brainClusterTable c[]; }

type fullBrainData {}
type fullBrainSpecs {}
type brainDataset {}

// Procedure to run "R" statistical package
(brainDataset t) bricRInvoke (script permutationScript, int iterationNo,
    brainMeasurements dataAll, precomputedPermutations dataPerm) {
       app { bricRInvoke @filename(permutationScript) iterationNo
                             @filename(dataAll) @filename(dataPerm); }
}

// Procedure to run AFNI Clustering tool
(brainClusterTable v, brainDataset t) bricCluster (script clusterScript,
  int iterationNo, brainDataset randBrain, fullBrainData brainFile,
  fullBrainSpecs specFile) {
       app { bricPerlCluster @filename(clusterScript) iterationNo
                            @filename(randBrain) @filename(brainFile)
                            @filename(specFile); }
}

// Procedure to merge results based on statistical likelhoods
(brainClusterTable t) bricCentralize ( brainClusterTable bc[]) {
      app { bricCentralize @filenames(bc); }
}
```

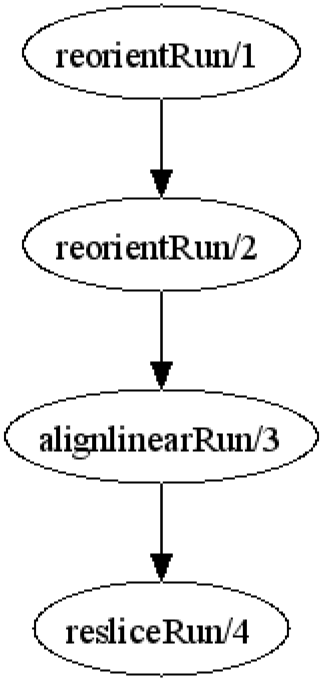
ACTIVAL: Dataset Iteration Procedures

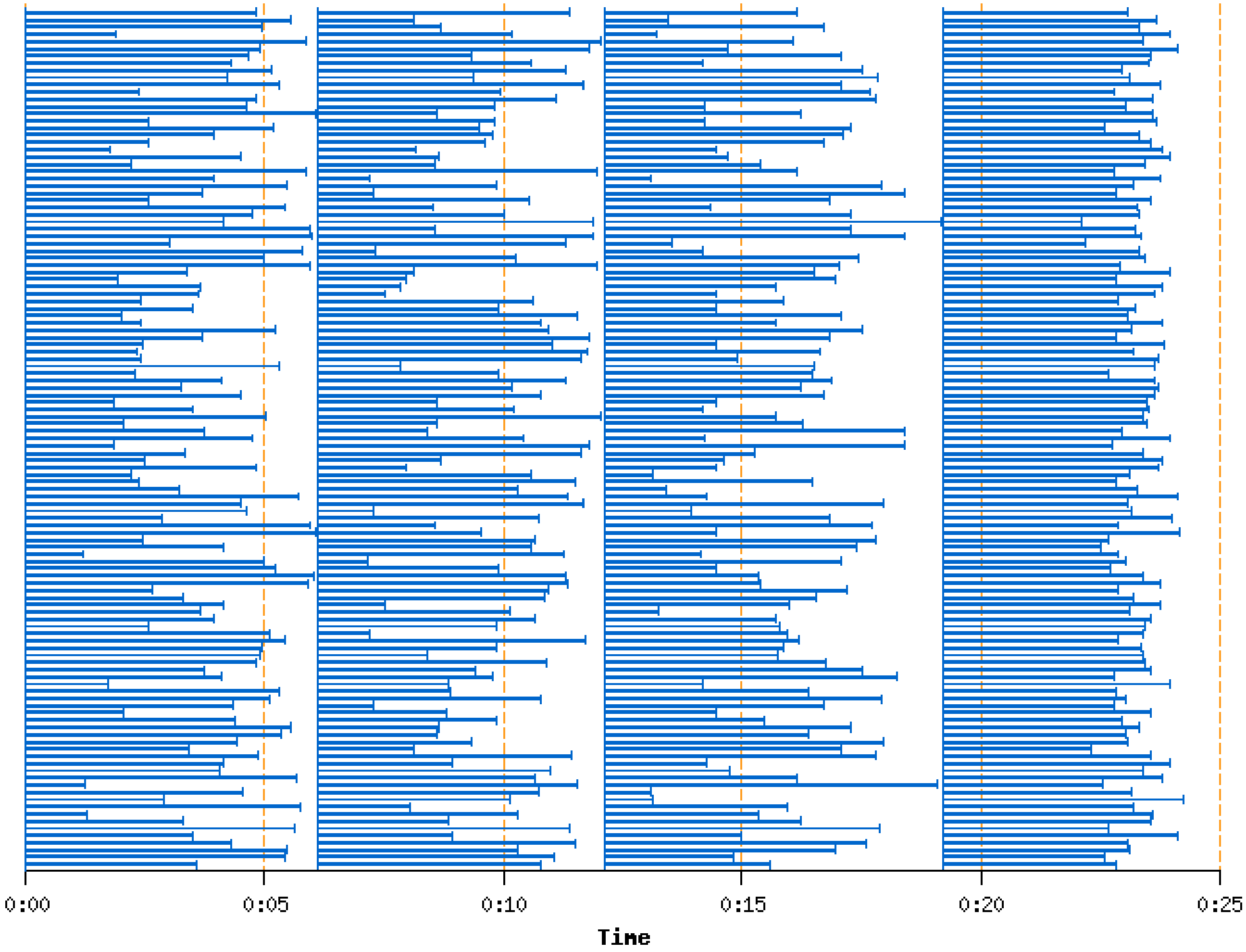
```
// Procedure to iterate over the data collection

(brainClusters randCluster, brainDatasets dsetReturn) brain_cluster
   (fullBrainData brainFile, fullBrainSpecs specFile)
{
   int sequence[]=[1:2000];

   brainMeasurements          dataAll<fixed_mapper; file="obs.imit.all">;
   precomputedPermutations dataPerm<fixed_mapper; file="perm.matrix.11">;
   script                             randScript<fixed_mapper; file="script.obs.imit.tibi">;
   script                             clusterScript<fixed_mapper; file="surfclust.tibi">;
   brainDatasets                   randBrains<simple_mapper; prefix="rand.brain.set">;

   foreach int i in sequence {
      randBrains.b[i] = bricRInvoke(randScript,i,dataAll,dataPerm);
      brainDataset rBrain=randBrains.b[i];
      (randCluster.c[i],dsetReturn.b[i]) =
         bricCluster(clusterScript,i,rBrain, brainFile,specFile);
   }
}
```

ACTIVAL: Main Program

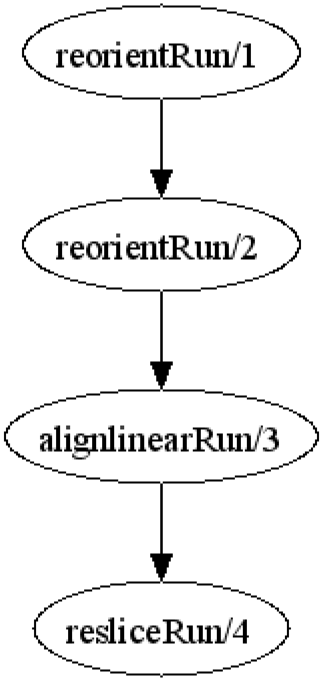
```
// Declare datasets

fullBrainData         brainFile<fixed_mapper; file="colin_lh_mesh140_std.pial.asc">;
fullBrainSpecs        specFile<fixed_mapper; file="colin_lh_mesh140_std.spec">;

brainDatasets         randBrain<simple_mapper; prefix="rand.brain.set">;
brainClusters         randCluster<simple_mapper; prefix="Tmean.4mm.perm",
                          suffix="_ClstTable_r4.1_a2.0.1D">;
brainDatasets         dsetReturn<simple_mapper; prefix="Tmean.4mm.perm",
                          suffix="_Clustered_r4.1_a2.0.niml.dset">;
brainClusterTable     clusterThresholdsTable<fixed_mapper; file="thresholds.table">;
brainDataset          brainResult<fixed_mapper; file="brain.final.dset">;
brainDataset          origBrain<fixed_mapper; file="brain.permutation.1">;


// Main program – executes the entire application

(randCluster, dsetReturn) = brain_cluster(brainFile, specFile);

clusterThresholdsTable = bricCentralize (randCluster.c);

brainResult = makebrain(origBrain,clusterThresholdsTable,brainFile,specFile);
```

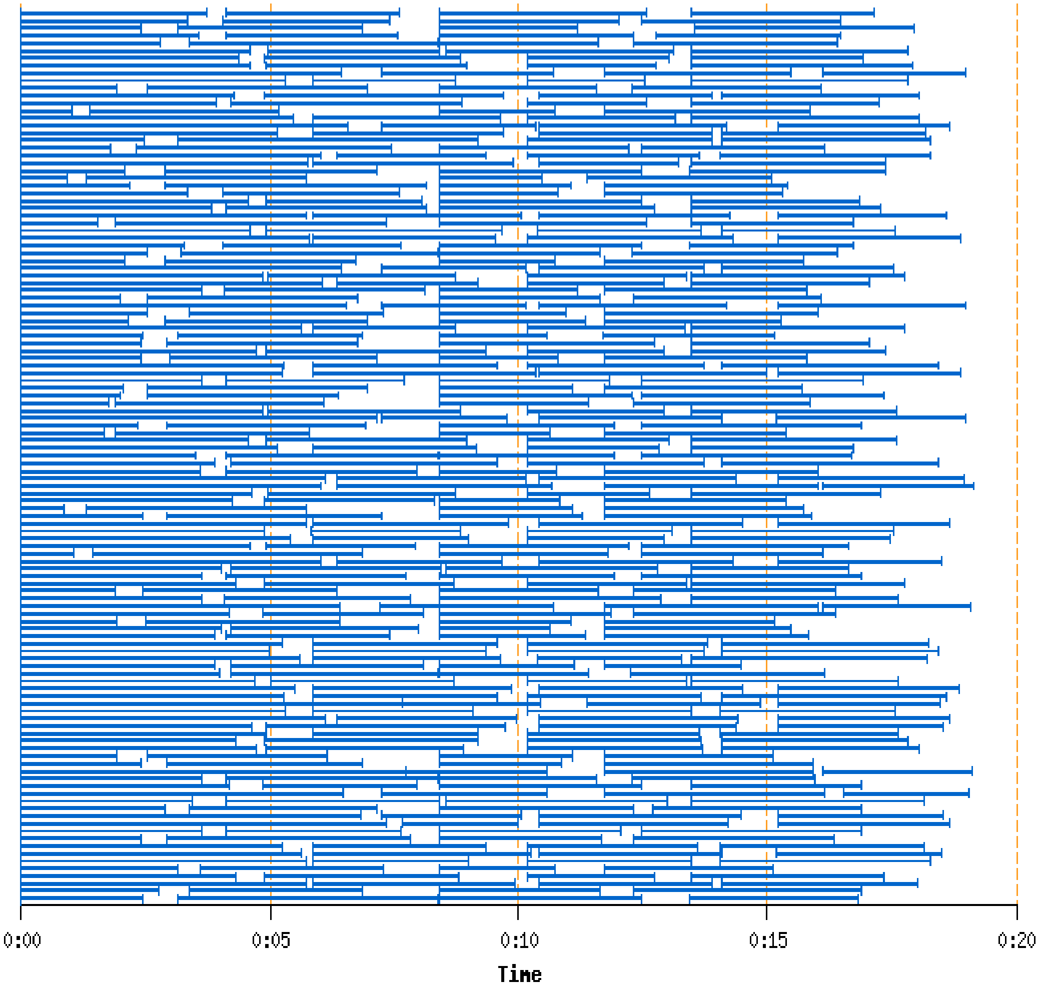
Example Performance Optimizations





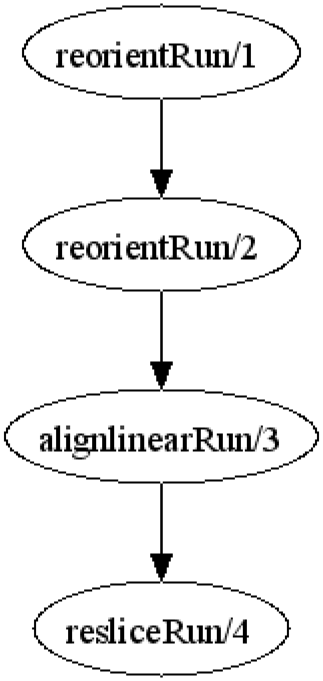
Example
Performance Optimizations





Pipelining

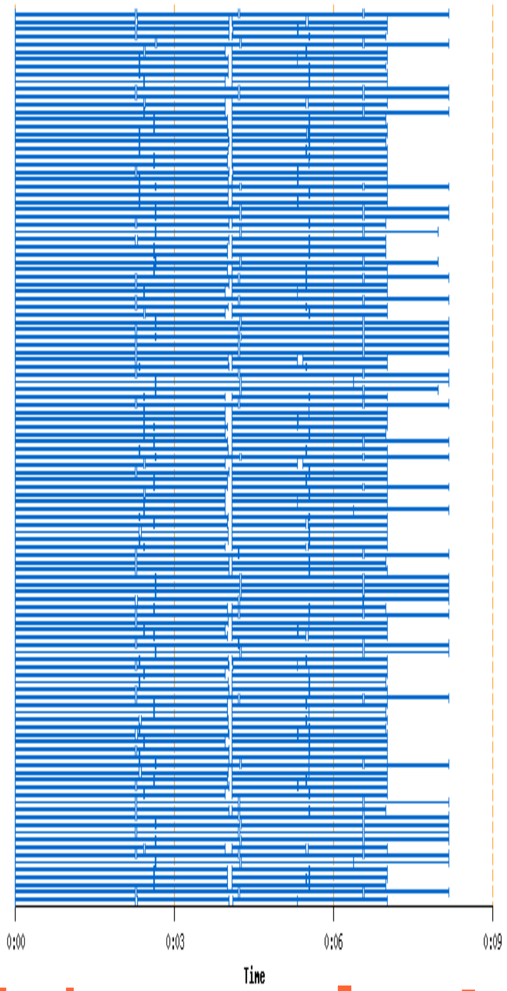
Example
Performance Optimizations


reorientRun/1

reorientRun/2

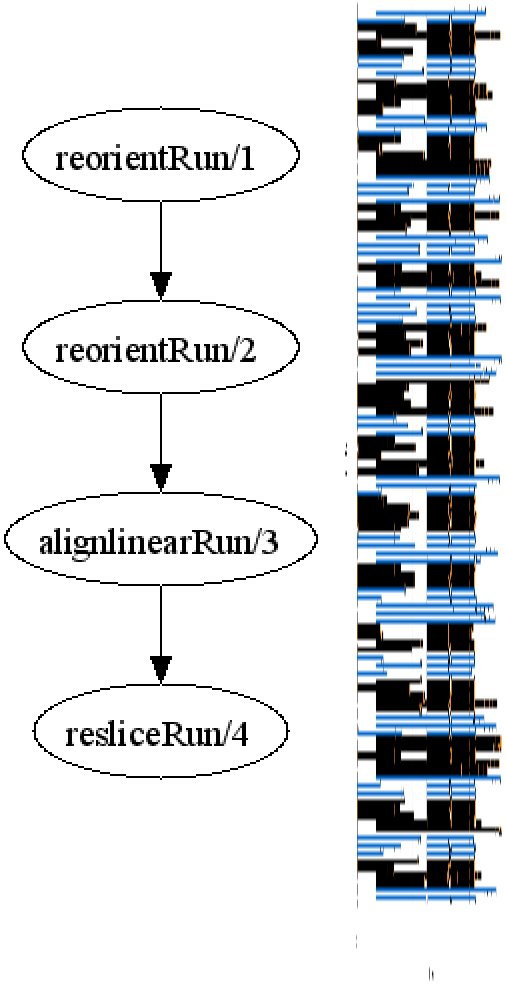
alignlinearRun/3

resliceRun/4



Pipelining + **clustering**

Example Performance Optimizations



Pipelining + **provisioning**

Other Applications

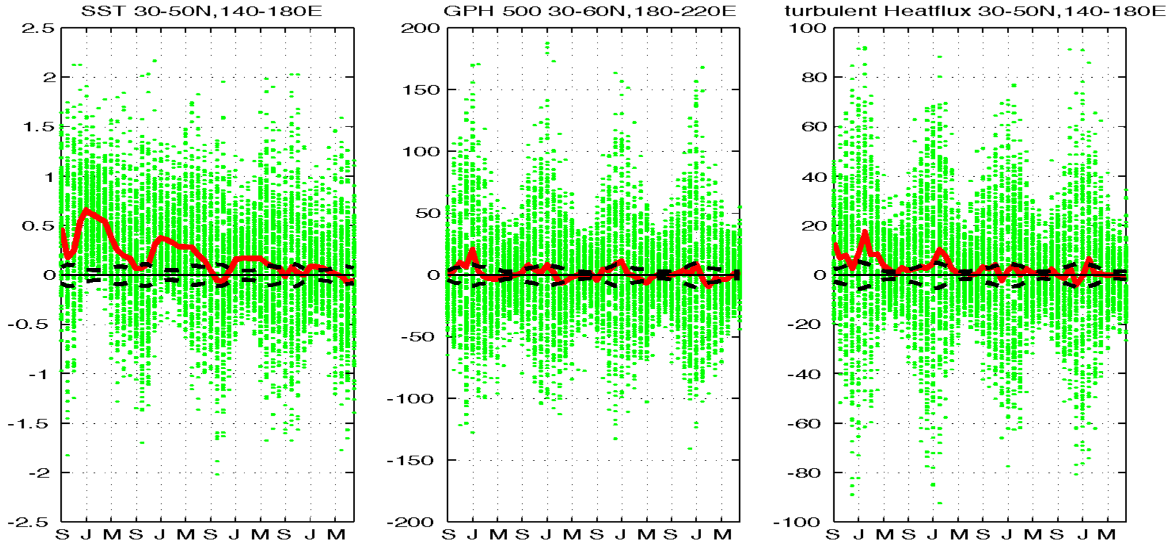
Application	#Jobs/computation	Levels
ATLAS* HEP Event Simulation	500K	1
fMRI DBIC* AIRSN Image Processing	100s	12
FOAM Ocean/Atmosphere Model	2000 (core app runs 250 8-CPU jobs)	3
GADU* Genomics: (14 million seq. analyzed)	40K	4
HNL fMRI Aphasia Study	500	4
NVO/NASA* Photorealistic Montage/Morphology	1000s	16
QuarkNet/I2U2* Physics Science Education	10s	3-6
RadCAD* Radiology Classifier Training	1000s	5
SIDGrid EEG Wavelet Proc, Gaze Analysis, …	100s	20
SDSS* Coadd, Cluster Search	40K, 500K	2, 8

Fast Ocean Atmosphere Model

NCAR

Manual config, execution, bookkeeping

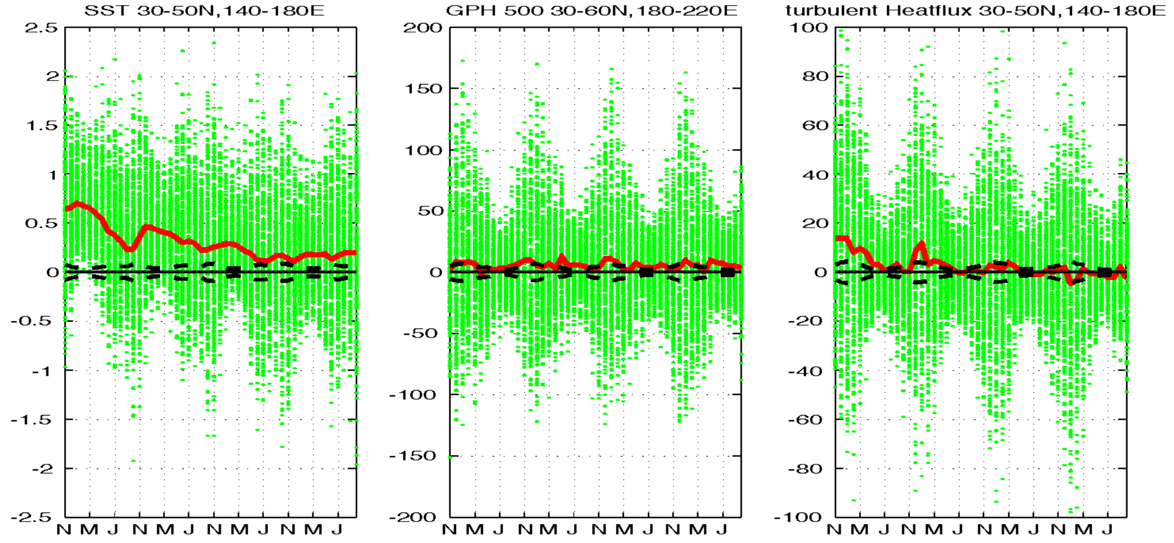
160 ensemble members = 2.5 months to run



VDS on Teragrid

Automated

250 ensemble members = 4 days to run



Green: each ensemble Red: ensemble mean

Visualization courtesy Pat Behling and Yun Liu, UW Madison

Swift System
www.ci.uchicago.edu/swift

- Clean separation of logical/physical concerns
  - ◆ **XDTM** specification of logical data structures

+ Concise specification of parallel programs
  - ◆ **SwiftScript**, with iteration, etc.

+ Efficient execution on distributed resources
  - ◆ Lightweight threading, dynamic provisioning, Grid interfaces, pipelining, load balancing

+ Rigorous provenance tracking and query
  - ◆ Virtual data schema & automated recording

→ **Improved usability and productivity**
  - ◆ Demonstrated in numerous applications