Dataflow Schedule Optimization on the Cloud

Yannis Ioannidis

MaDgIK Lab

University of Athens & ATHENA Research Center

Joint Work with ...

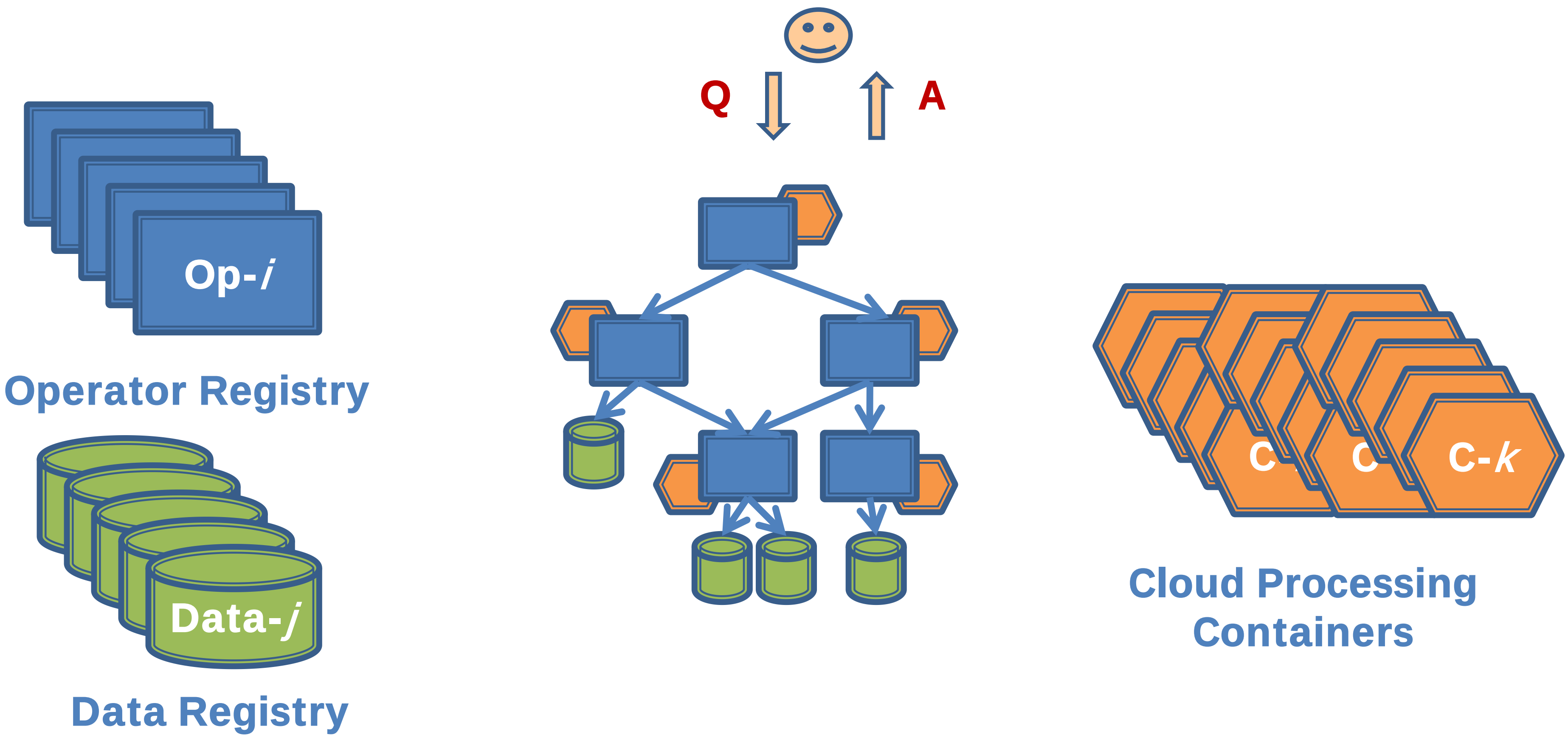
- Herald Kllapi
- Eva Sitaridi
- Manolis Tsangaris
- ...
- Dimitris Achlioptas (future)

Contents

- Challenges
- Motivation
- The ADP System
- Problem definition
- Approach
- Experimental evaluation
- Conclusions & Future work

Challenges

Querying/Analysis/Processing on a Data Infrastructure



Challenge Focus

- Big Data Processing
  - TB or PB of data (scientific, sensors, ...)
  - Efficiency
- High-level Data Languages
  - Languages to easily express data operations
  - Semantics
- (Query) optimization
  - Reconciling efficiency and semantics

Big Data Processing Systems

- Hadoop
  - Open source software for reliable, scalable, distributed computing
  - Won Jim Gray's Terabyte Sort Benchmark in 2008 (209 seconds)
- Google Map-Reduce
  - Jim Gray's Terabyte Sort Benchmark in 68 seconds in 2009
- PNUTS (Yahoo! Research)
  - Massively parallel & geographically distributed database system
- Pegasus
  - Scientific workflows on the Grid
- Dryad (Microsoft Research)
  - General-purpose distributed execution engine for coarse-grain data-parallel applications

High-level Languages

- Hive-QL
  - SQL-Like
- Pig-Latin
  - Dataflow language
- Mashups
  - Yahoo! pipes
  - MashQL

High-level Languages (Hive-QL)

- Hive-QL is based on SQL

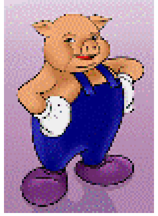
```
CREATE TABLE page_view(
        viewTime INT,
        userid BIGINT,
        page_url STRING,
        referrer_url STRING,
        ip STRING COMMENT 'IP Address of the User')
COMMENT 'This is the page view table'
PARTITIONED BY(
        dt STRING,
        country STRING)
STORED AS SEQUENCEFILE;
```

Create tables

Write queries

```
INSERT OVERWRITE TABLE xyz_com_page_views
SELECT page_views.*
FROM page_views
WHERE page_views.date >= '2008-03-01'
        AND page_views.date <= '2008-03-31'
        AND page_views.referrer_url like '%xyz.com';
```

High-level Languages (Pig-Latin)

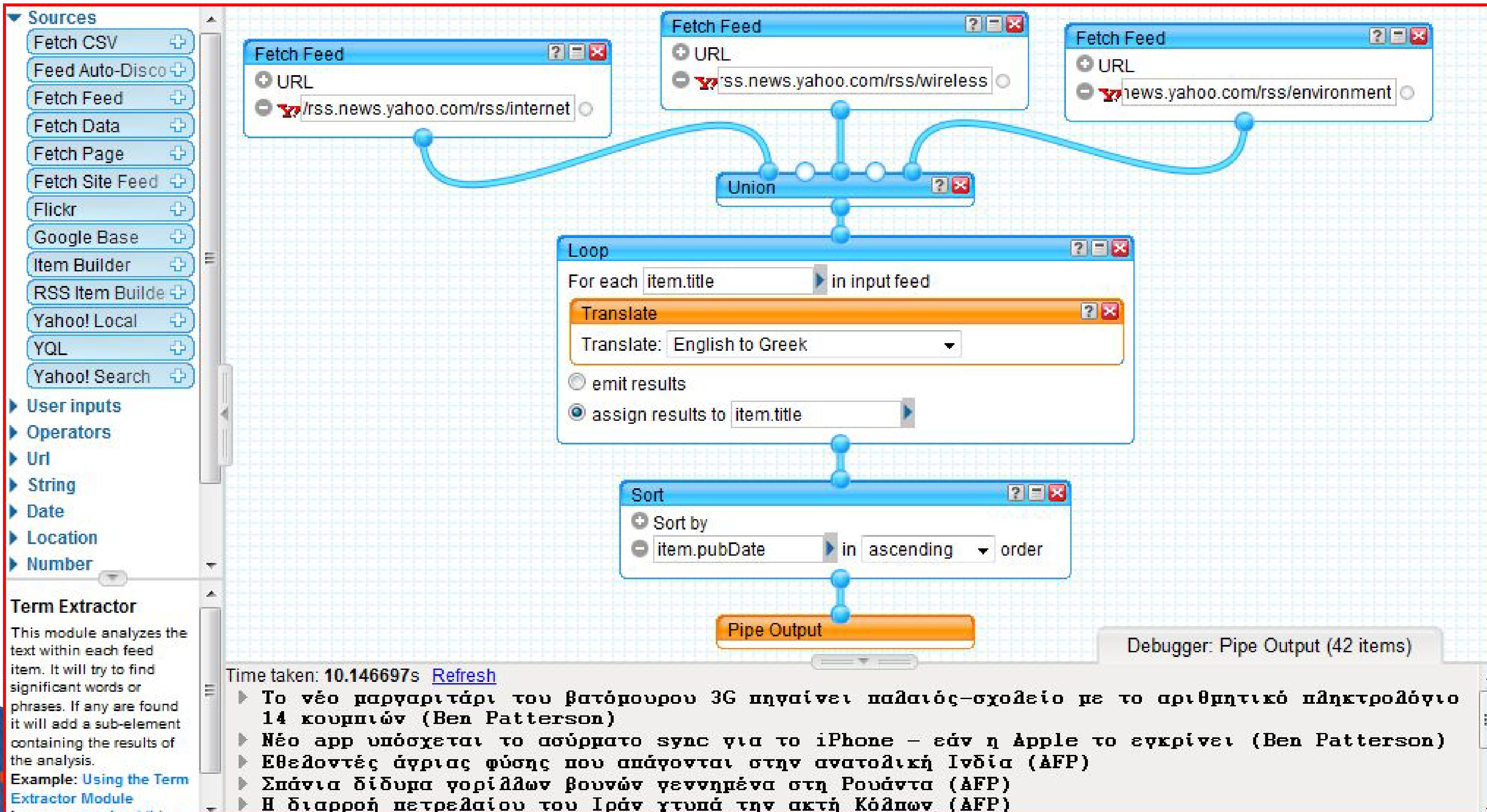


- Pig-Latin is a dataflow language

```
SET default_parallel 20;
A = LOAD 'myfile.txt' USING PigStorage() AS (t, u, v);
B = GROUP A BY t;
C = FOREACH B GENERATE group, COUNT(A.t) as mycount;
D = ORDER C BY mycount;
STORE D INTO 'mysortedcount' USING PigStorage();
```

High-level Languages (Yahoo! Pipes)

- Graphical mashup builder from Yahoo!

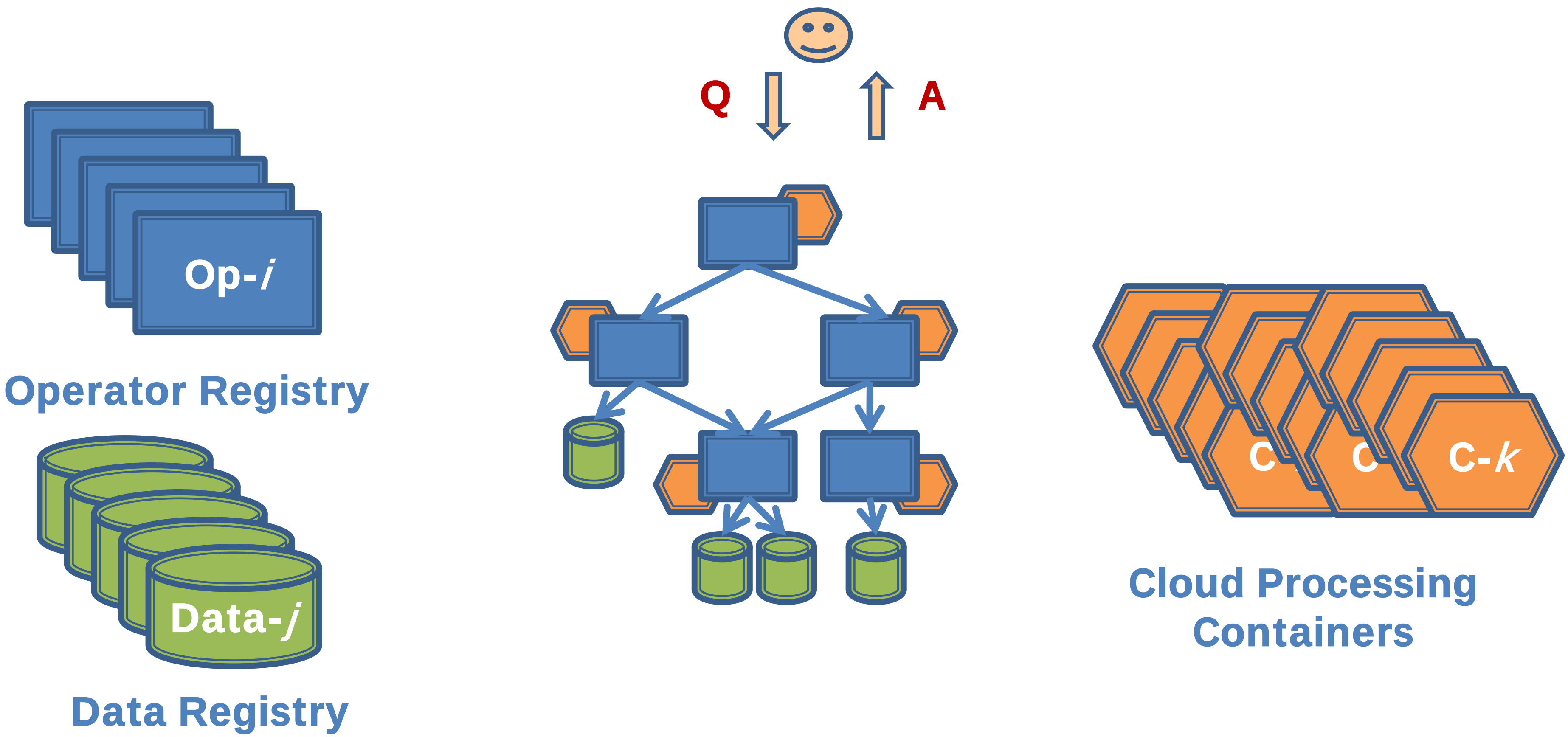


Optimization

- Hadoop!
  - Push the operation as close to the data as possible
- Condor
  - Designed for CPU intensive applications
  - Matchmaking with ClassAds
- Pegasus
  - Uses condor for scheduling

Motivation

Querying/Analysis/Processing on a Data Infrastructure



Classical Query Optimization

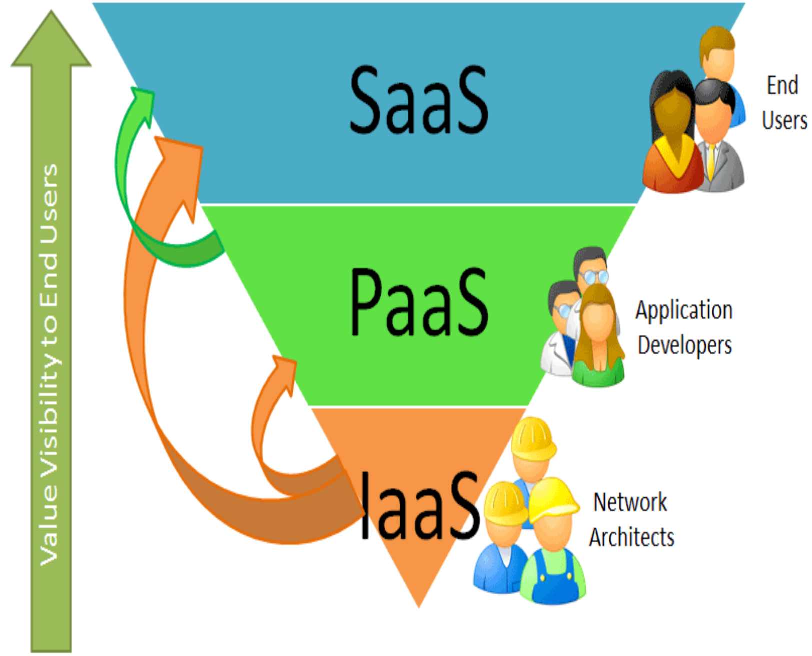
- Query: graph of relational algebra operators
- Optimality: response time or completion time
- Environment: cluster of dedicated distributed / parallel hosts

Emerging Query Optimization

- Query: graph of arbitrary operators
- Optimality: response time or completion time <u>and</u> money
- Environment: cloud of hosts (elasticity)

Cloud Computing 101

- Virtualized IT resources offered as on-demand service
  - Software as a Service (IaaS)
  - Platform as a Service (PaaS)
  - Infrastructure as a Service (SaaS)



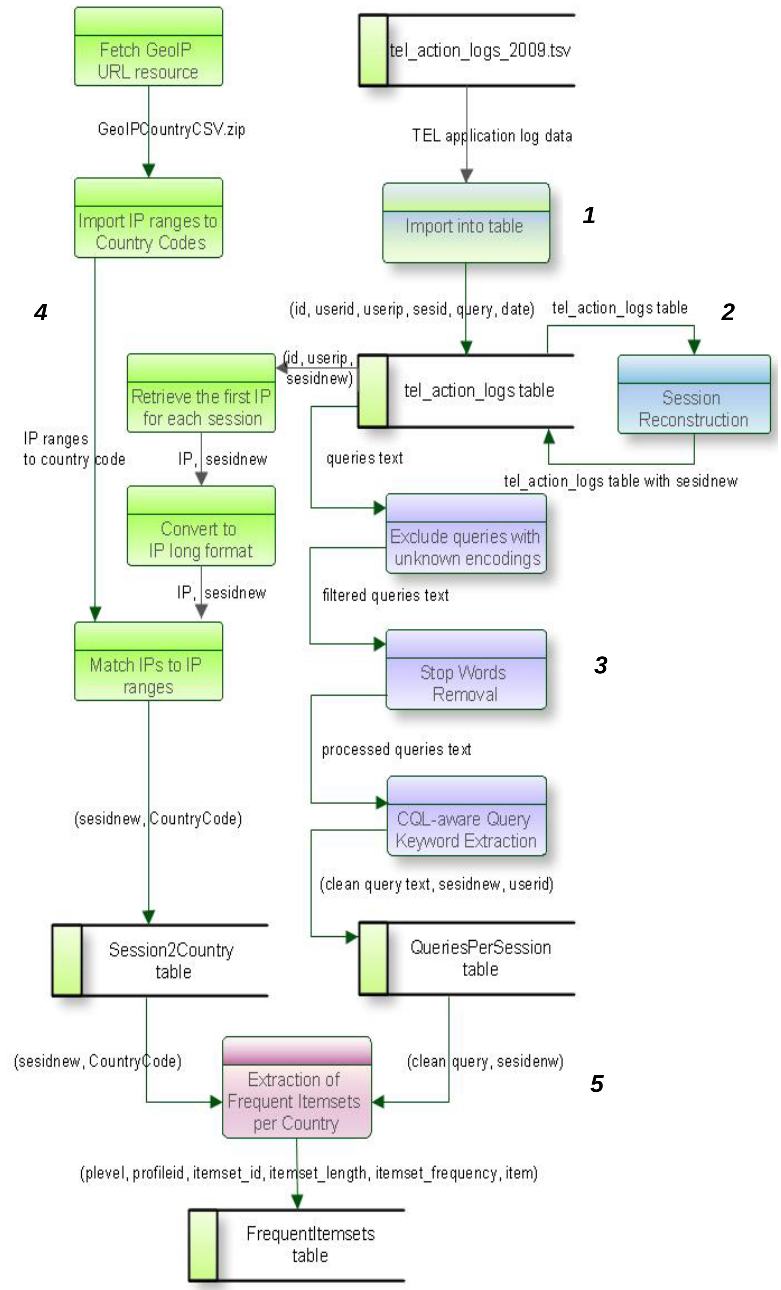
- Variety of charging and use policies

Cloud Computing 101

- Cloud of hosts (elasticity)
- Virtual resources (virtual hosts = containers)
  - Available on demand
  - Used for as much time needed
  - Leased on a per quantum pricing scheme
- Illusion of infinite resources
- Arbitrary # of choices of price/performance ratio

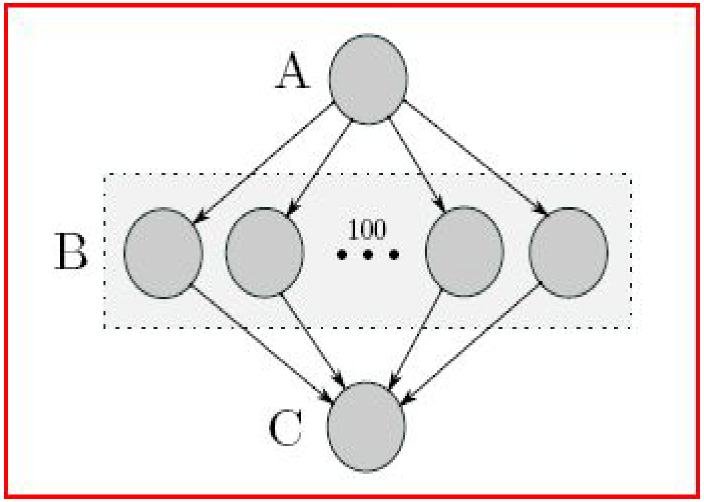
Motivation

- Graph of arbitrary operators
- Non-relational data analytics
  - Query log analysis
  - Data mining
  - Simulation model composition
  - ...
- User behavior analysis for European national libraries
  - One of sixteen flows



Motivation: Elasticity/Tradeoff

- Time and money
- 2-dimensional optimization
- **Quantum: 1 hour**

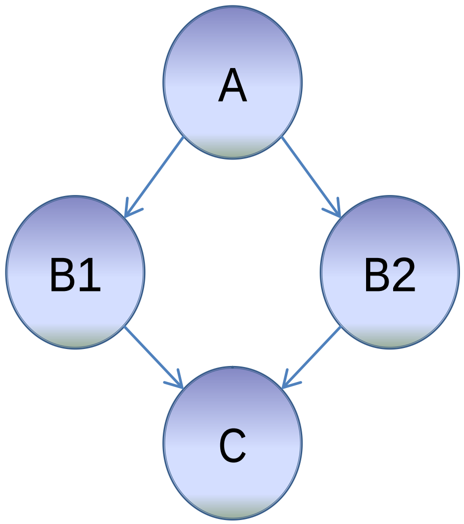


- Simple map-reduce flow
  - A: 1 hour B: 10 minutes C: 1 hour

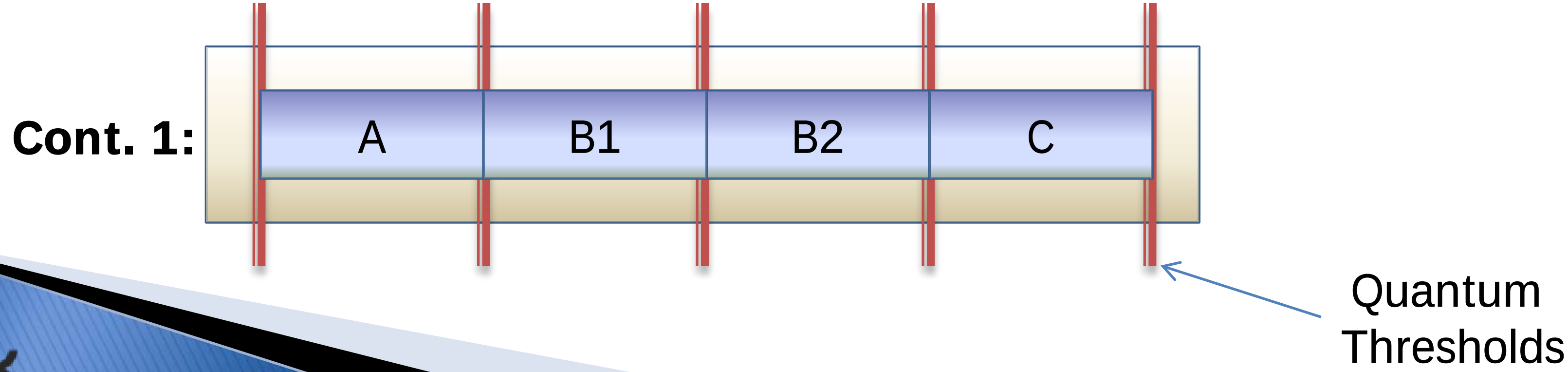
Schedule	Time (hours)	Money (resource hours)	Winner
One host for all ops	18.60	**19**	5x cheaper
Different host per op	**2.16**	102	9x faster

Motivation: Data Size/Net Speed

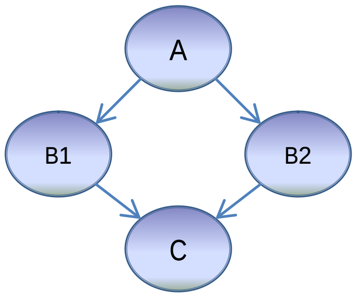
- Simple map-reduce flow with 1 split (A), 2 maps (B1, B2), and 1 reduce (C)
- A, B1, B2, C: 1 hour
- **Quantum: 1 hour**



Schedule	Time (hours)	Money (resource hours)
One host for all ops	4.00	**4**

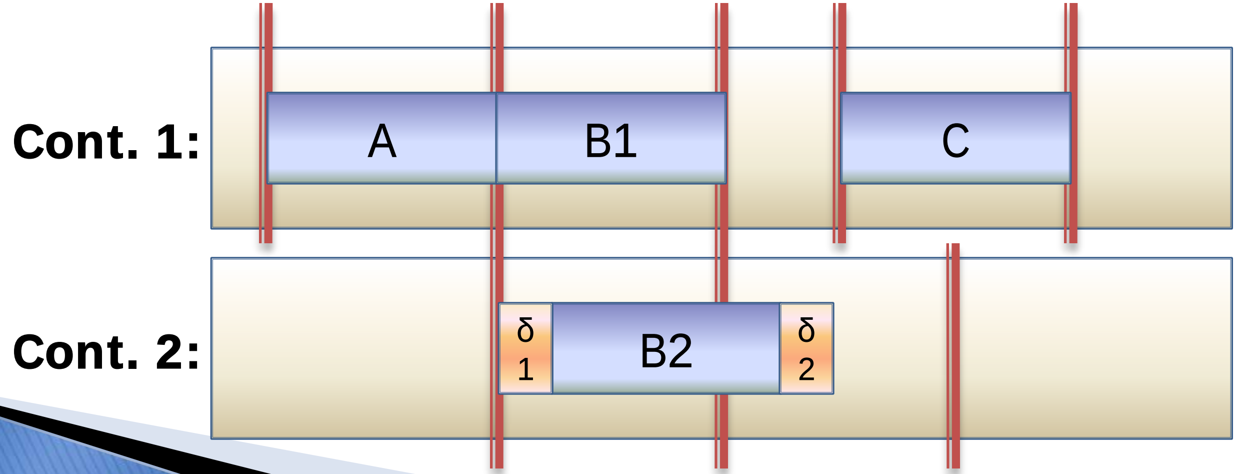


Motivation: Data Size/Net Speed



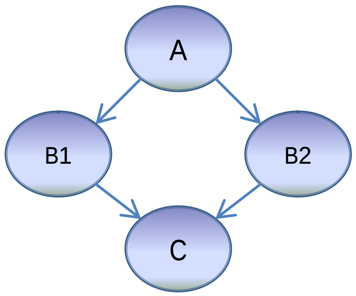
- Small output

Schedule	Time (hours)	Money (resource hours)
One host for all ops	4.00	**4**
Two hosts, small output	3.50	**5**



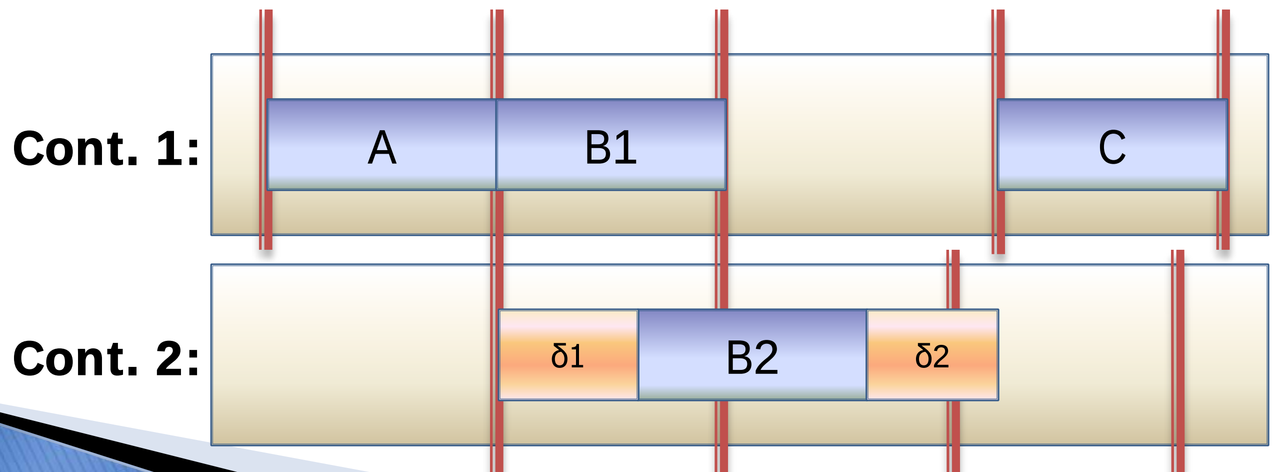
$\delta1 + \delta2 = 0.5$

Motivation: Data Size/Net Speed



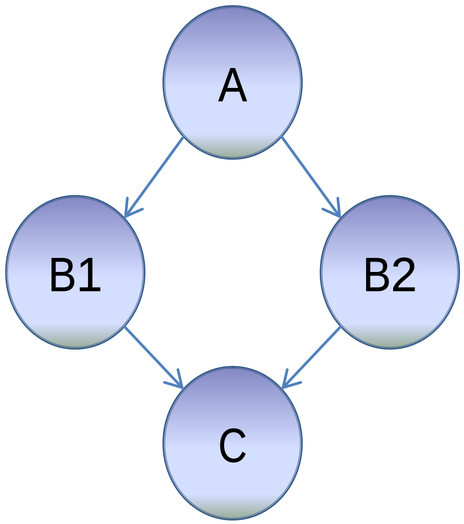
- Large output

Schedule	Time (hours)	Money (resource hours)
One host for all ops	4.00	**4**
Two hosts, small output	3.50	**5**
Two hosts, large output	4.20	**6**



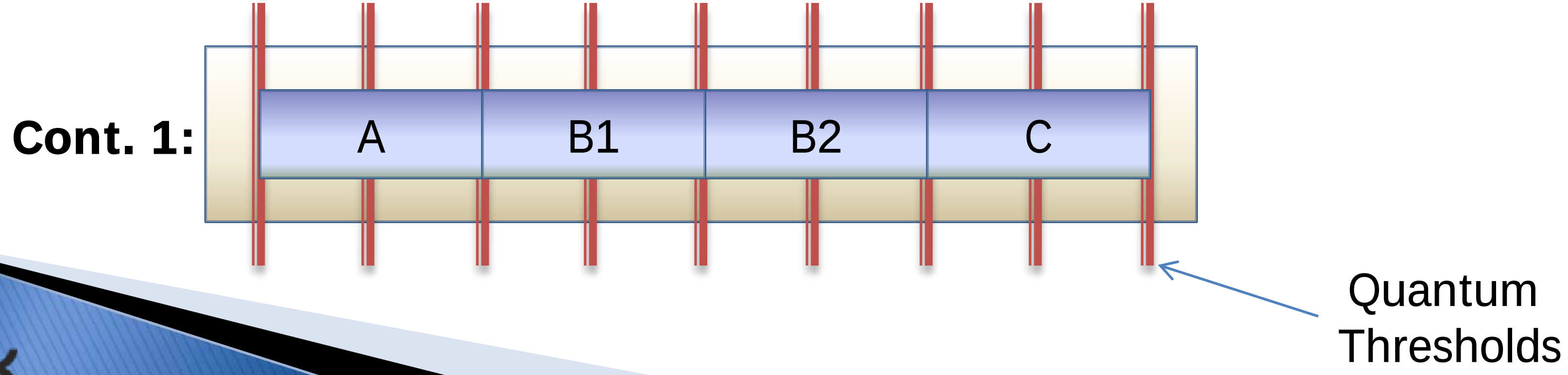
$\delta 1 + \delta 2 = 1.2$

Motivation: Charging Policies

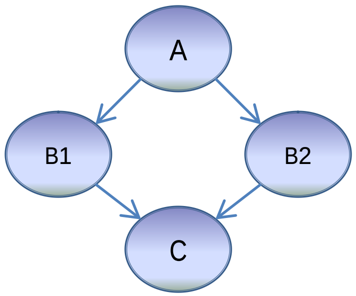


- Simple map-reduce flow with 1 split (A), 2 maps (B1, B2), and 1 reduce (C)
- A, B1, B2, C: 1 hour
- **Quantum: 0.5 hours**

Schedule	Time (hours)	Money (resource hours)
One host for all ops	4.00	**4.0**

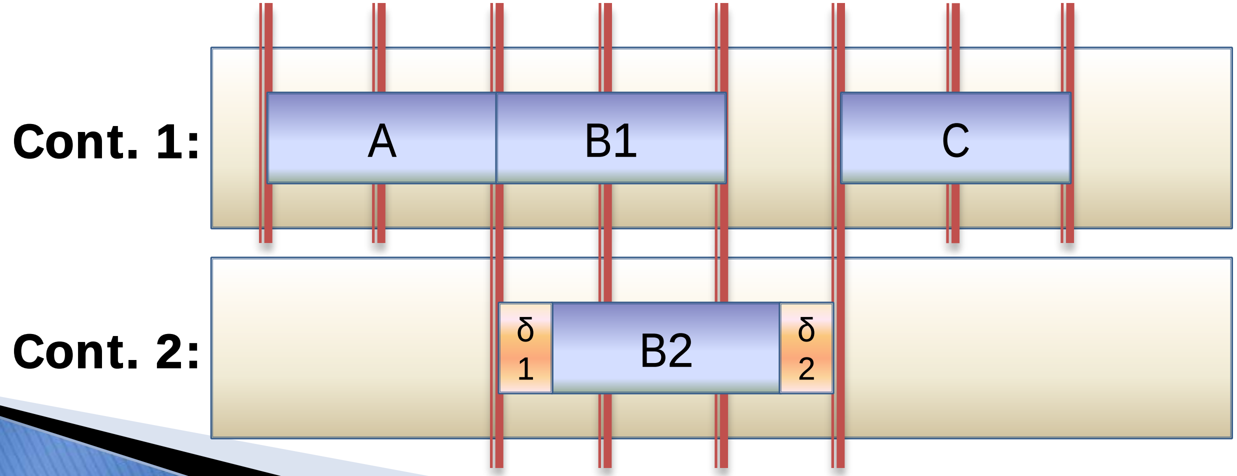


Motivation: Charging Policies



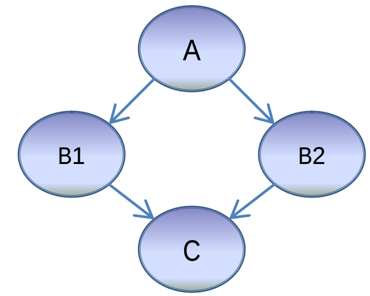
- Small output

Schedule	Time (hours)	Money (resource hours)
One host for all ops	4.00	**4.0**
Two hosts, q=0.5 hour	3.50	**4.5**
Two hosts, q=1.0 hour	3.50	**5.0**



$\delta 1 + \delta 2 = 0.5$

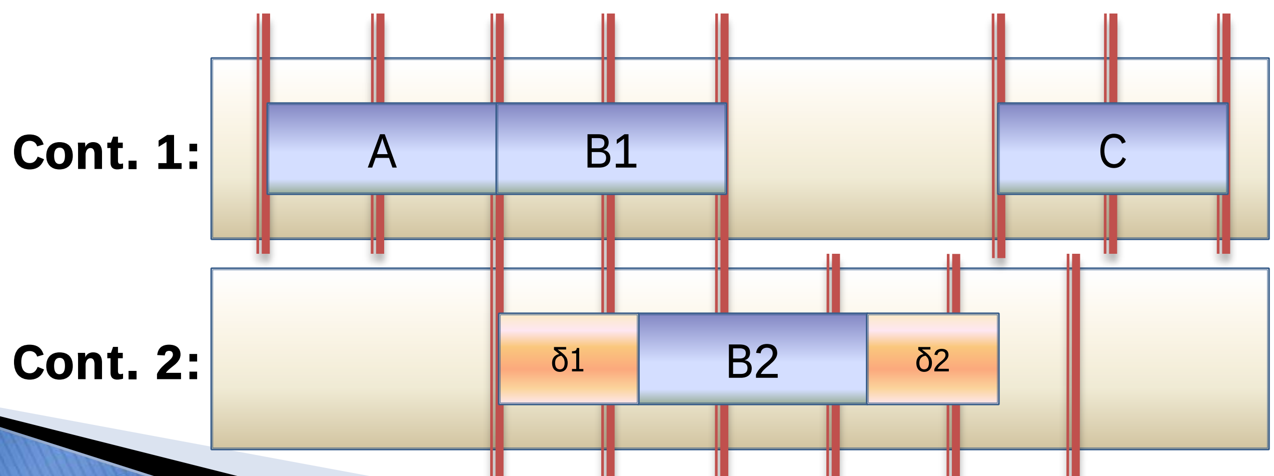
Motivation: Charging Policies



- Large output

Schedule	Time (hours)	Money (resource hours)
One host for all ops	4.00	**4.0**
Two hosts, q=0.5 hour	4.20	**5.5**
Two hosts, q=1.0 hour	4.20	**6.0**

Dominated by

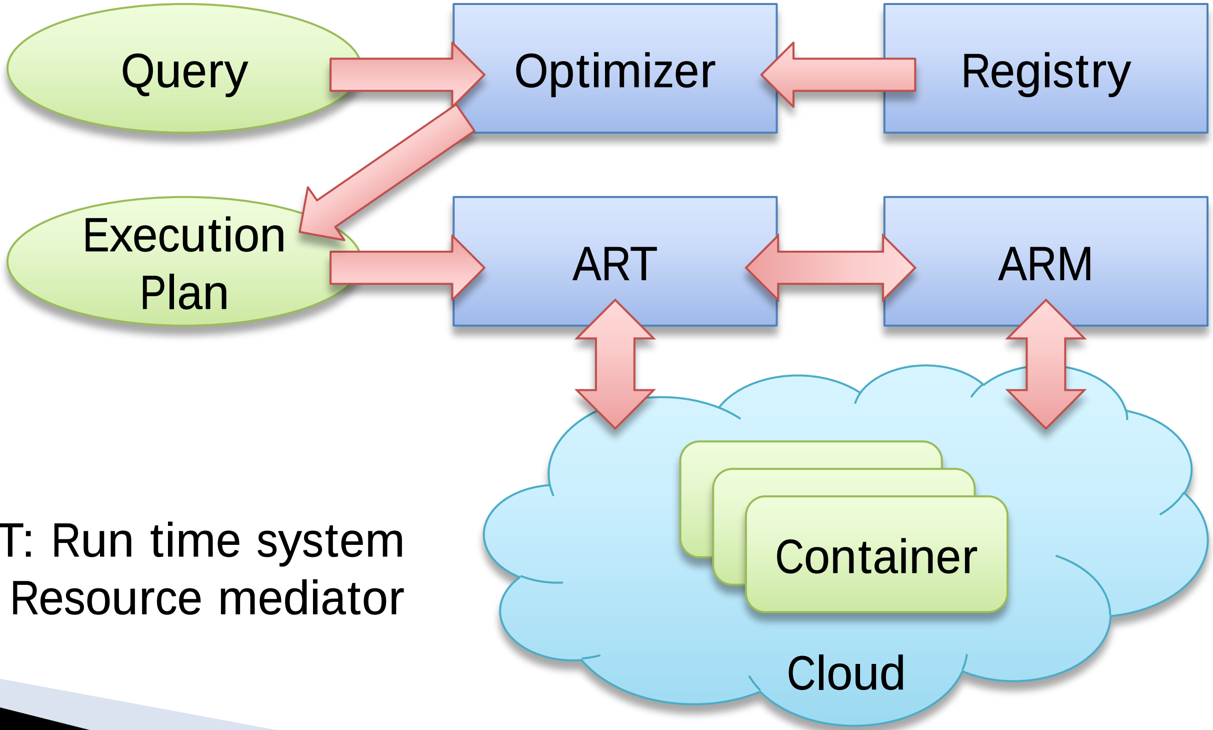


$\delta 1 + \delta 2 = 1.2$

The ADP System

The ADP System

- Athens Distributed Processing System
- Dataflow processing & optimization
- High-level queries transformed into dataflow graphs



ART: Run time system
ARM: Resource mediator

Optimization Challenges

- Variety of parameters
  - Monetary cost of resources
  - Freshness of data
  - ...
- Ad-hoc operators
  - Behavior is not known a-priori
- Variety of environments
  - Clusters
  - Clouds
  - ...
- Huge space of alternatives

Query Optimization in ADP

- Queries represented in three abstraction levels
  - Operator Graphs → Algebraic operators
  - Concrete Operator Graphs → Software operators
  - Execution Plans → Hosted operators
- Huge space of alternatives
  - Optimization performed in three corresponding steps
  - Different choices at every step

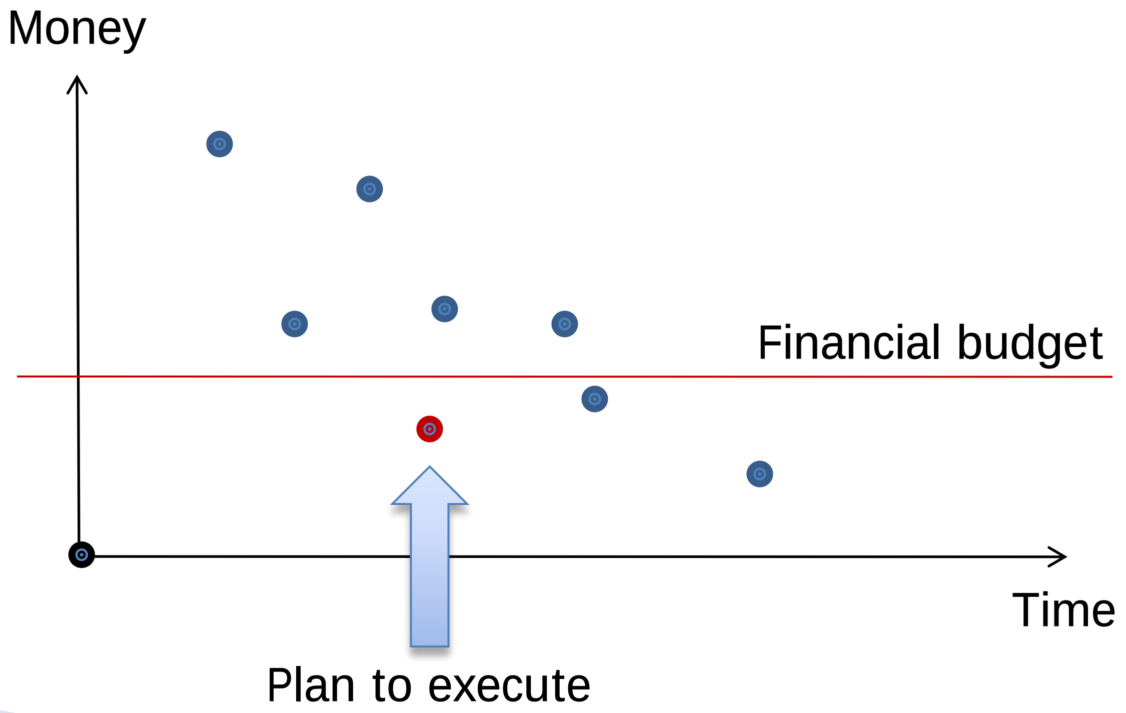
Problem Definition

Optimal Dataflow Scheduling

- Dataflow scheduling (execution plan derivation)
  - on the cloud with elastic resources
  - optimizing tradeoff between completion time & money
    - possibly constrained
    - possibly left to the user
  - of arbitrary operators with known characteristics

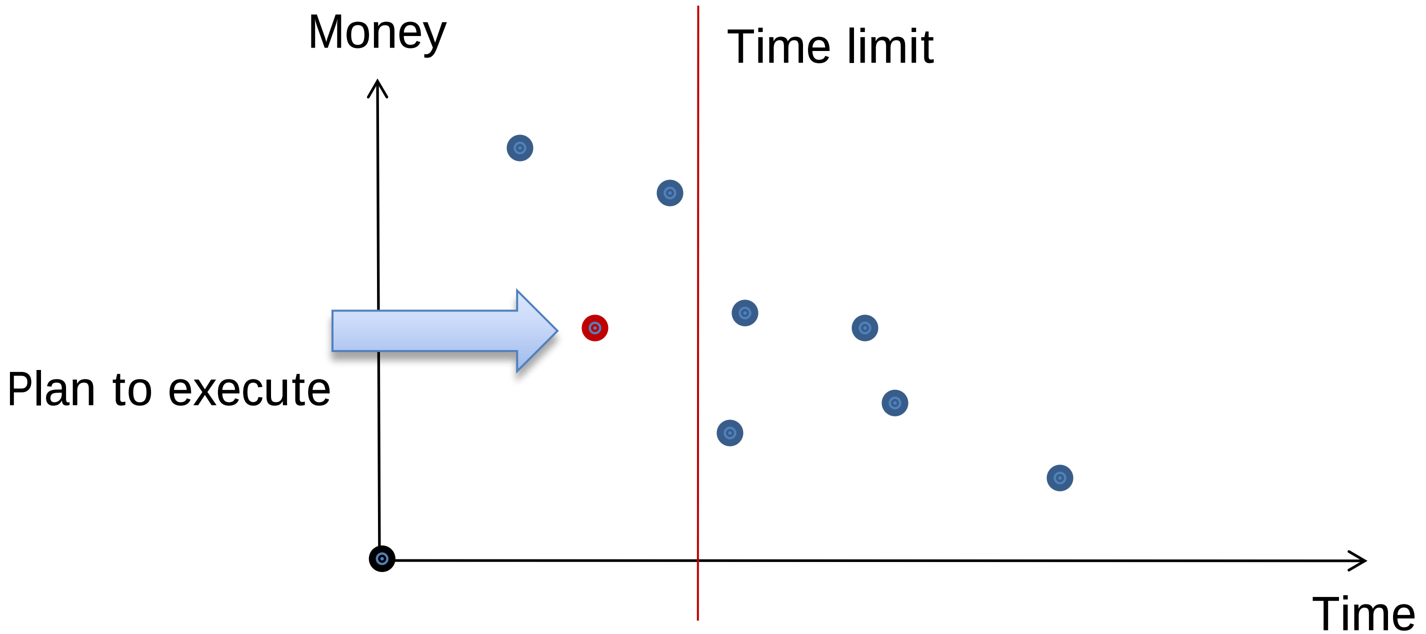
Optimal Dataflow Scheduling

- Fastest plan within specific financial budget



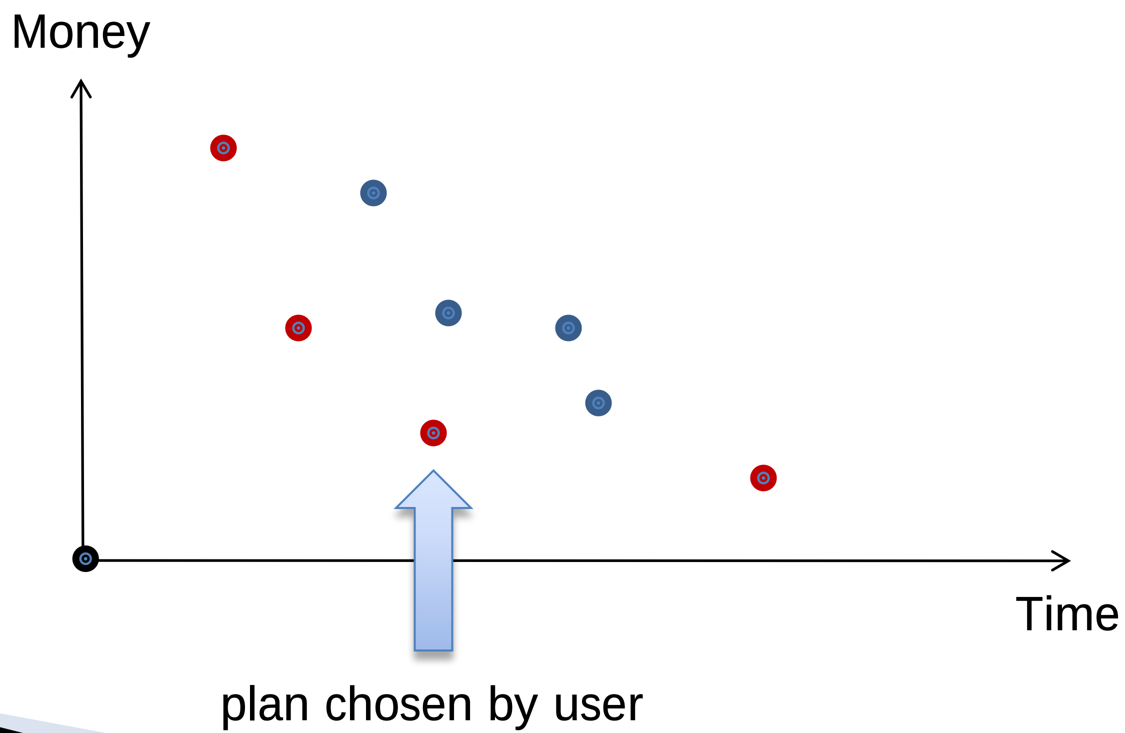
Optimal Dataflow Scheduling

- Cheapest plan within specific time limit



Optimal Dataflow Scheduling

- Skyline of all Pareto optimal plans

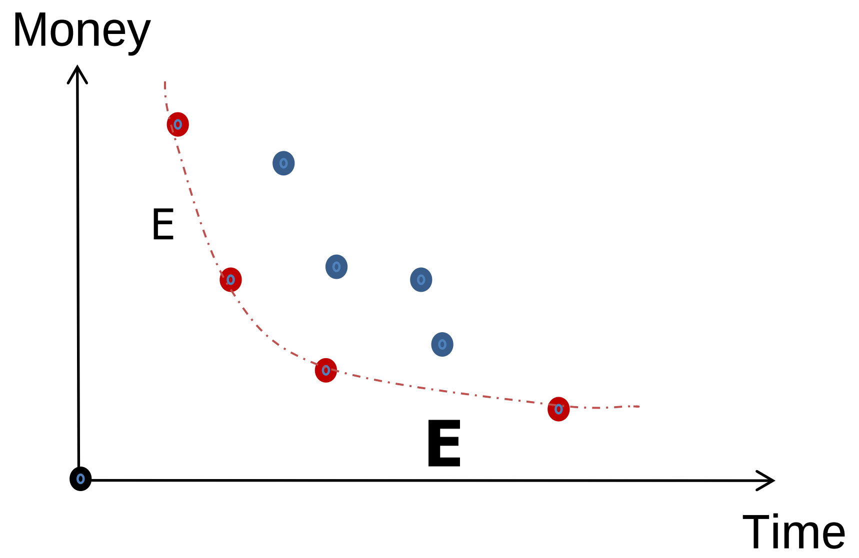


Optimal Dataflow Scheduling

- Constrained problems are symmetric
- Constrained problems: user provides time limits or budgets before optimization
- Skyline problem: user chooses best tradeoff after optimization

Dataflow Elasticity

- Speed of completion time reduction when more money is available



$$E = \frac{(T_{max} - T_{min})/T_{max}}{(M_{max} - M_{min})/M_{max}}$$

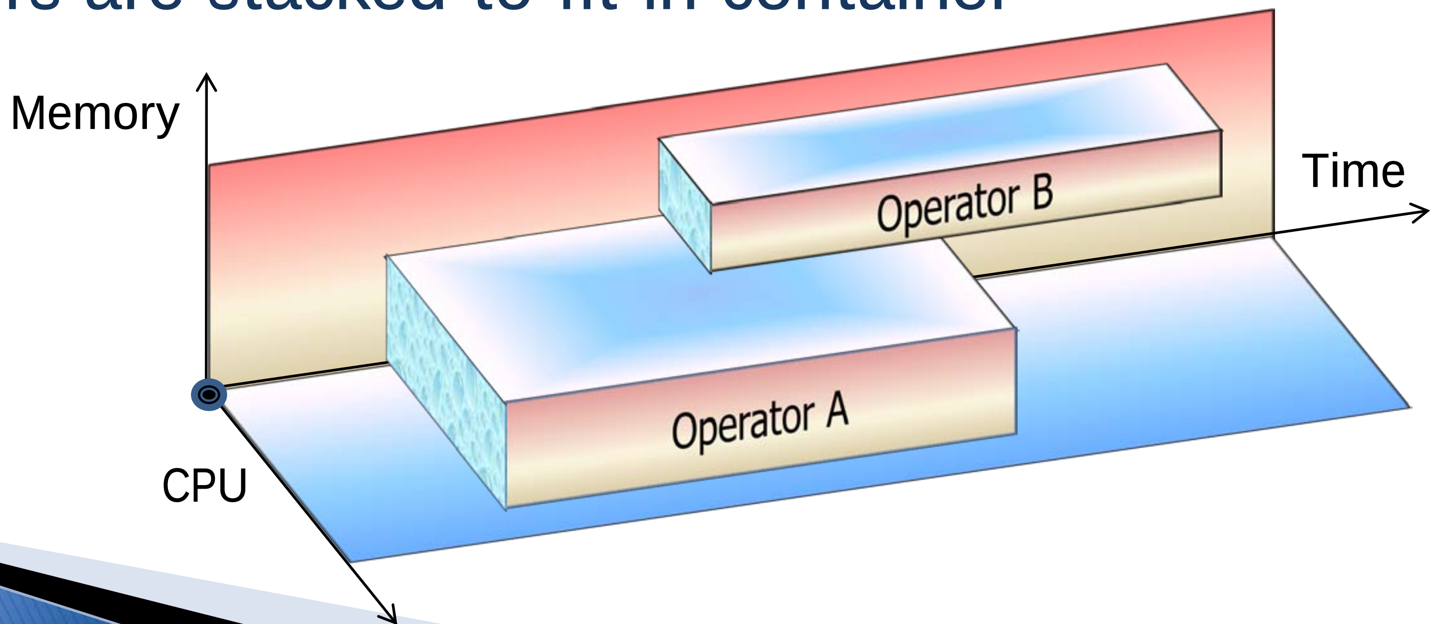
Approach

Dataflow, Operator, & Container Modeling

- Dataflow: graph(ops, flows)
- Operator: op(time, cpu, memory, behavior)
  - time: completion time
  - cpu: CPU utilization (e.g., 80%)
  - memory: maximum memory required
  - behavior: **pipeline** or **store-and-forward**
    - *Select* is pipeline, *Sort* is store-and-forward
- Flow: flow(producer, consumer, data)
- Container: cont(cpu, memory, network)
  - network: input/output rate (e.g., 100 MB/sec)

Intuitive Representation

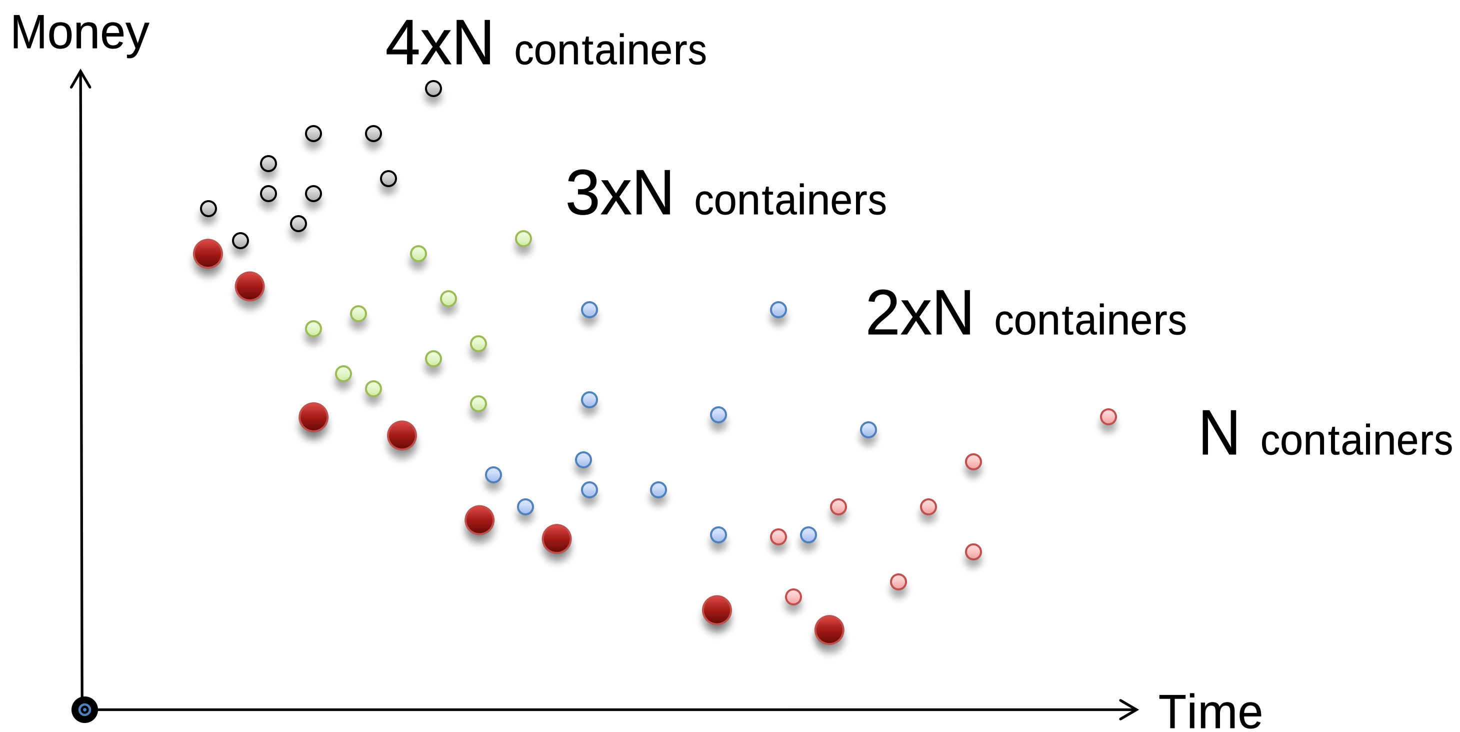
- Simplified 3D representation: CPU, memory, time
  - Operator: box of resource requirements
  - Container: empty box of CPU & memory capacities and infinite time
- Operators are stacked to fit in container



Optimization Constraints

- Space-shared resources (memory)
  - Hard constraints to be satisfied for operators to run
- Time-shared resources (cpu, network)
  - Can be multiplexed at the expense of time
- Dataflow constraints for consumers
  - Store-and-forward: Wait until all inputs are ready
  - Pipeline: Wait until store-and-forward inputs are ready

Optimization Alg Abstraction



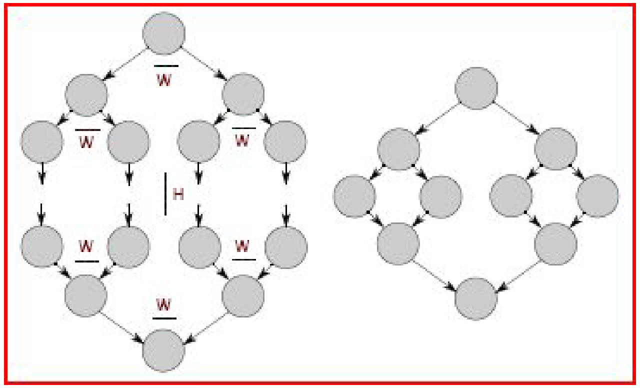
Experimental Evaluation

Experimental Testbed

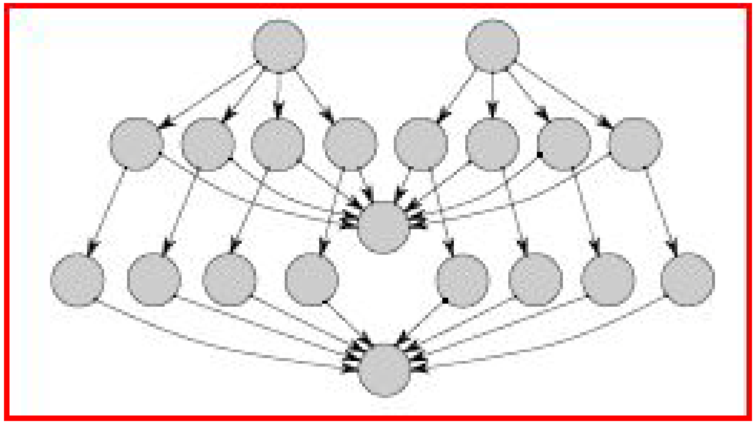
- Dataflow graph
  - Lattice, Ligo, Montage, CyberShake
  - Approximately 500 operators
- Operators
  - 100% store-and-forward
  - 100% pipeline
- Scheduling method
  - All algorithms
- Execution Environment
  - Different output data sizes
  - Multi- & Uni- Processing

Dataflow Graphs

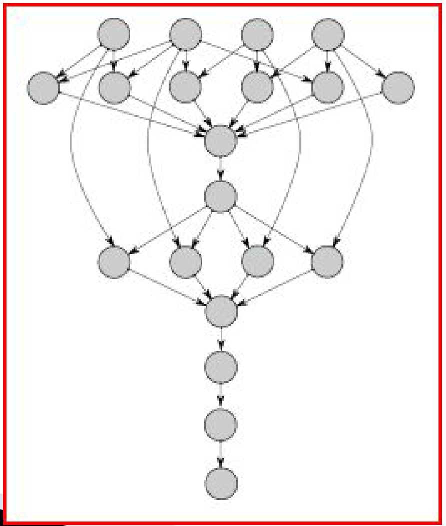
Lattice



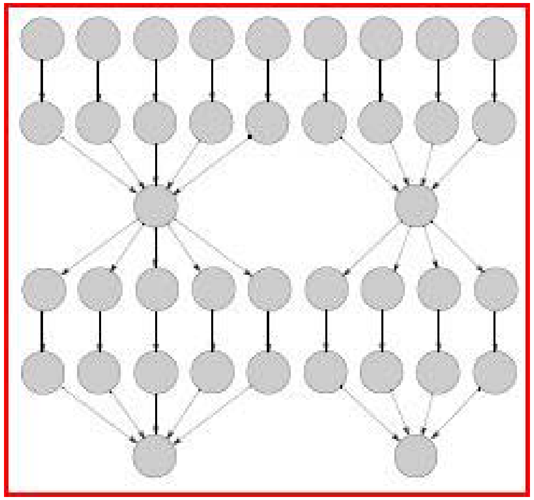
CyberShake



Montage



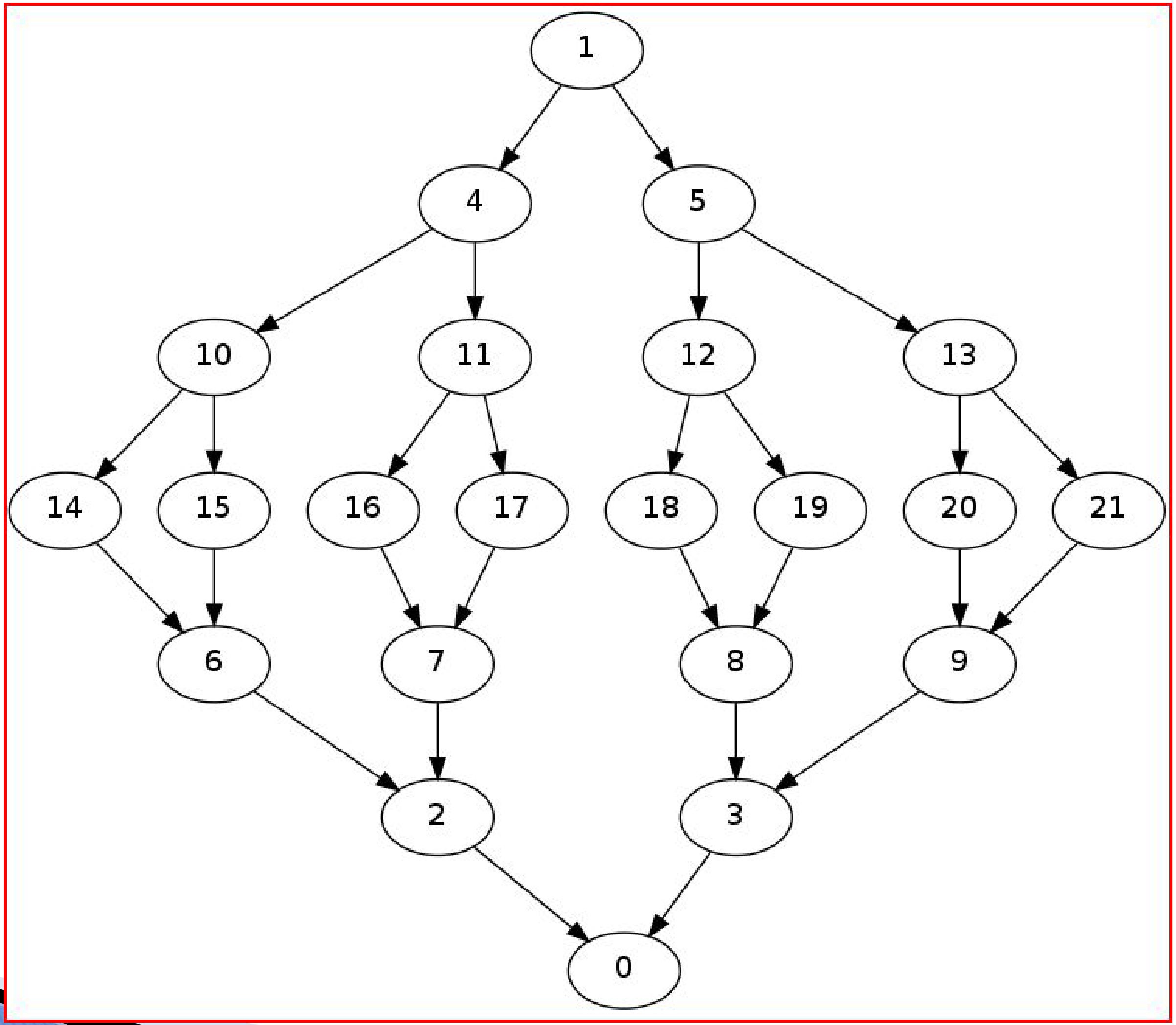
Ligo



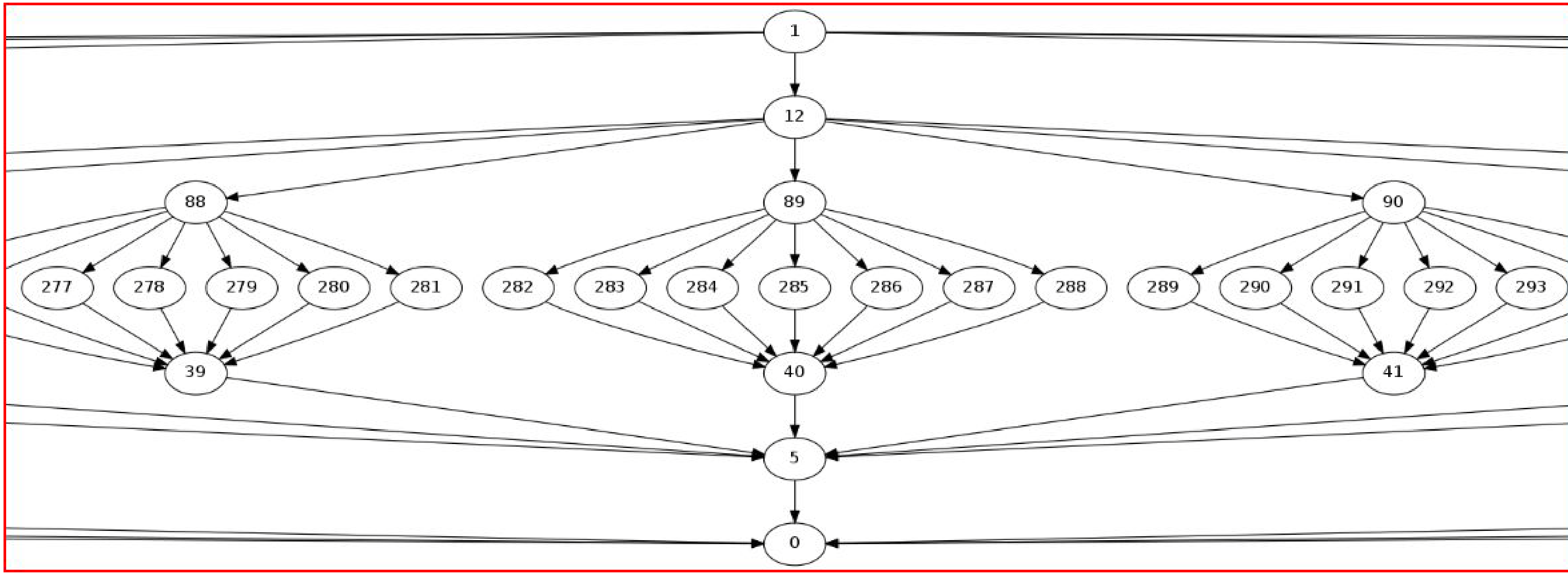
Dataflow Graphs

- Montage
  - Created by NASA/IPAC
  - Used to generate custom mosaics of the sky
- Ligo
  - Used to analyze binary galactic systems
- CyberShake
  - Created by Southern Calfornia Earthquake Center
  - Used to characterize earthquakes
- Lattice
  - Generalized map-reduce
  - Height 3 → standard map-reduce

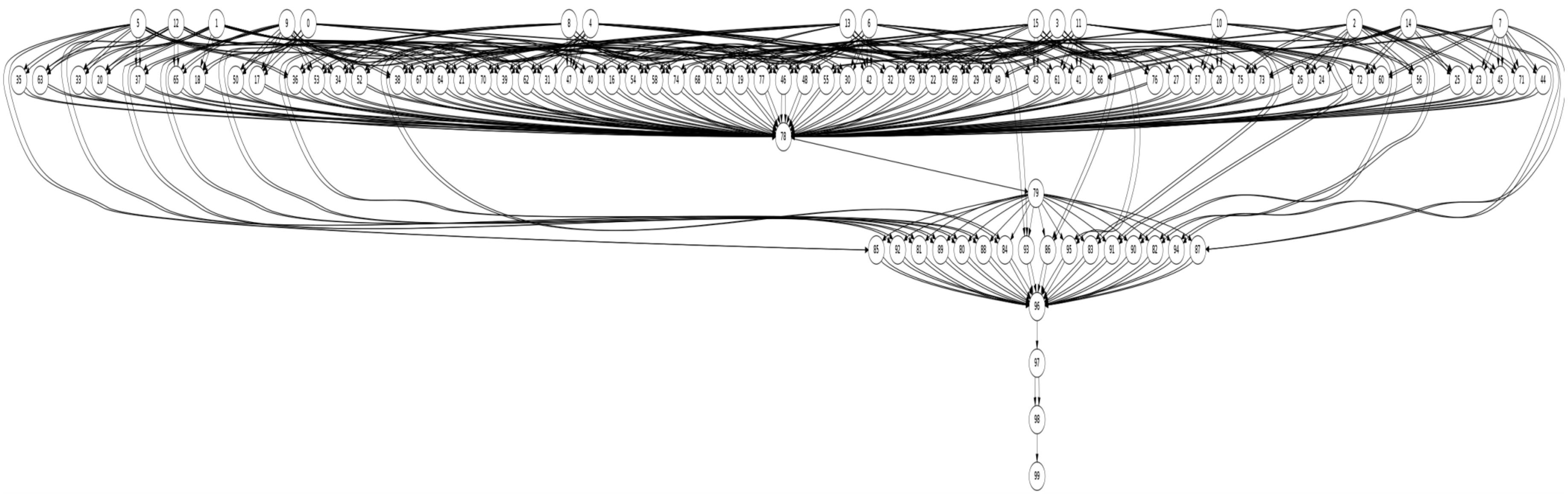
Lattice 7-2



Lattice 7-7
(A small part only)



Montage with 100 operators

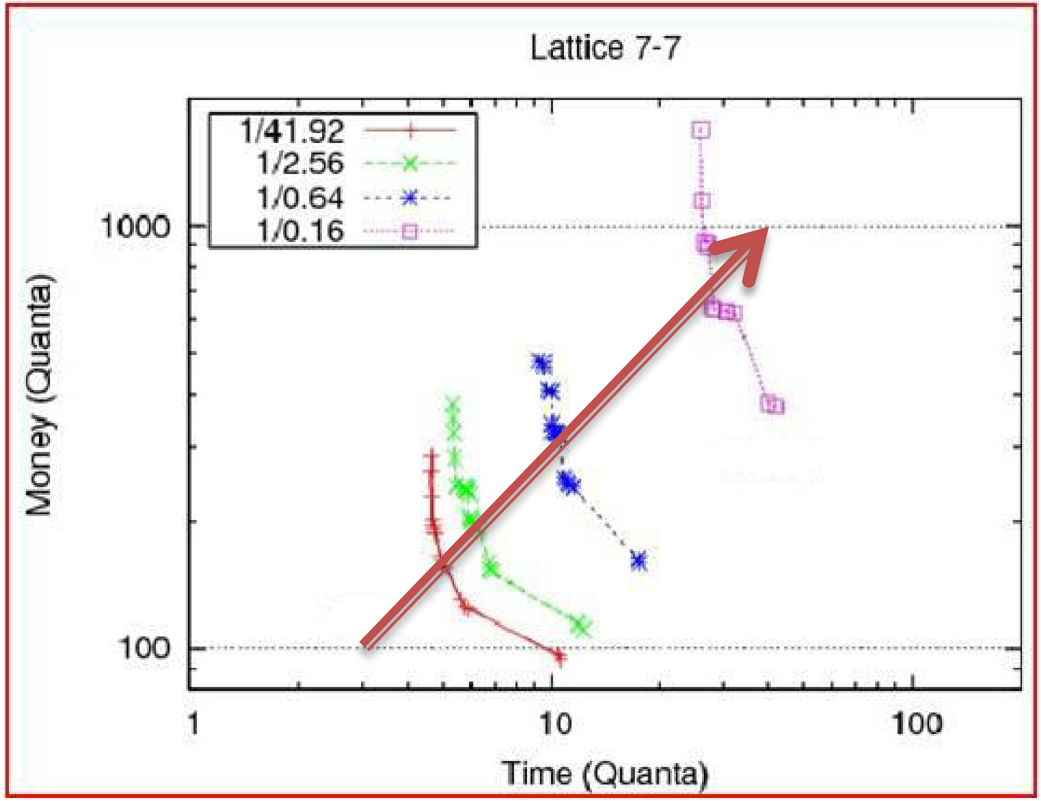


x5

Results
(Space Exploration)

Results

- Lattice 7-7
- 100% S&F
- 10.000 random plans
- Varying parameter
  - 10 – 150 containers
  - Output size



Large operator output size reduces elasticity

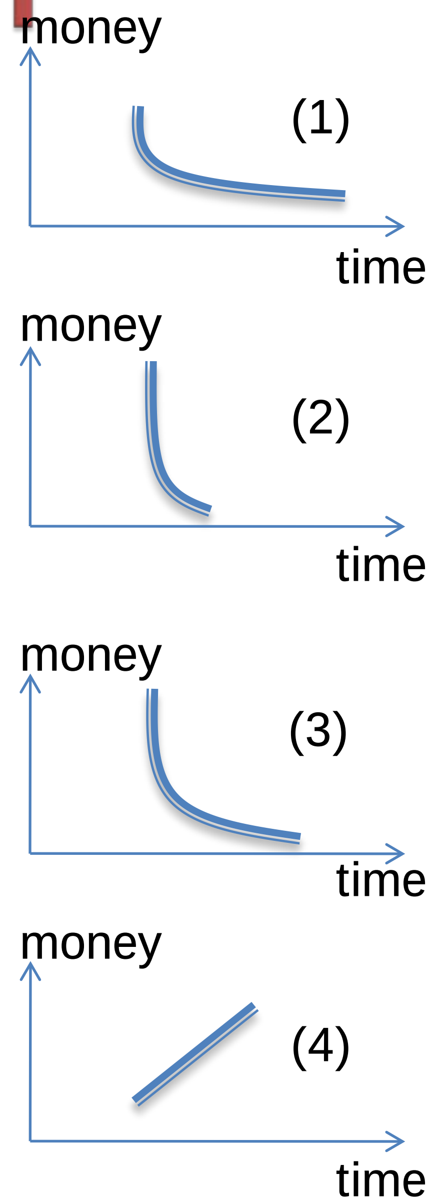
Results & Conclusion
(Optimization Algorithms)

Conclusions

- Different forms of elasticity depending on
  - type of the workload
  - network bandwidth/amount of data transferred
- Skyline contains plans by different algorithms
- Skylines of algorithms and space exploration close
- Simulated annealing does not improve significantly plans produced by some greedy algorithms

Preliminary Classification

- Very elastic plans (1)
  - Money has great impact on time
  - Low output and high graph parallelism
- Less elastic plans (2)
  - Money have little impact on time
  - Low output and low graph parallelism
- Average elasticity (3)
  - Balanced money/time tradeoff with knee
  - High output and high graph parallelism
- No elasticity (4)
  - Fastest plan is also cheapest
  - High output and low graph parallelism



THANK YOU!