TEXAS TECH UNIVERSITY

DISCL

Empowering Data-driven Discovery with a Lightweight Provenance Service for High Performance Computing

Yong Chen
Associate Professor, Computer Science Department
Director, Data-Intensive Scalable Computing Laboratory
Site Director, Cloud and Autonomic Computing Center
Texas Tech University

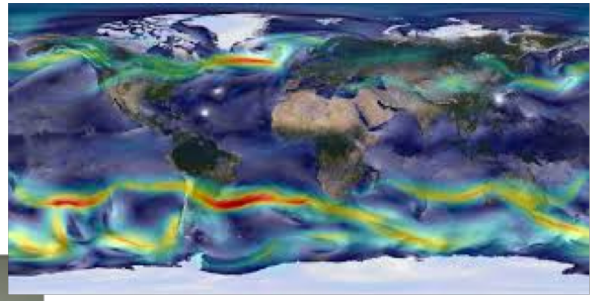
NITRD's MAGIC Webinar, April 3, 2019

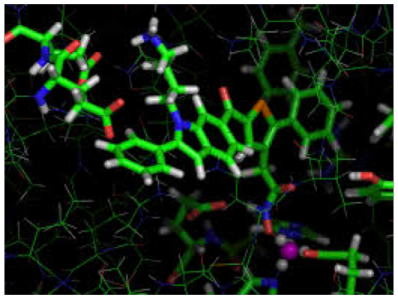
Data Challenges

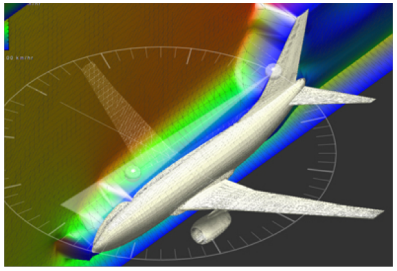
Scientific discovery becomes highly data intensive ("big data")

Both experimental data and observational data

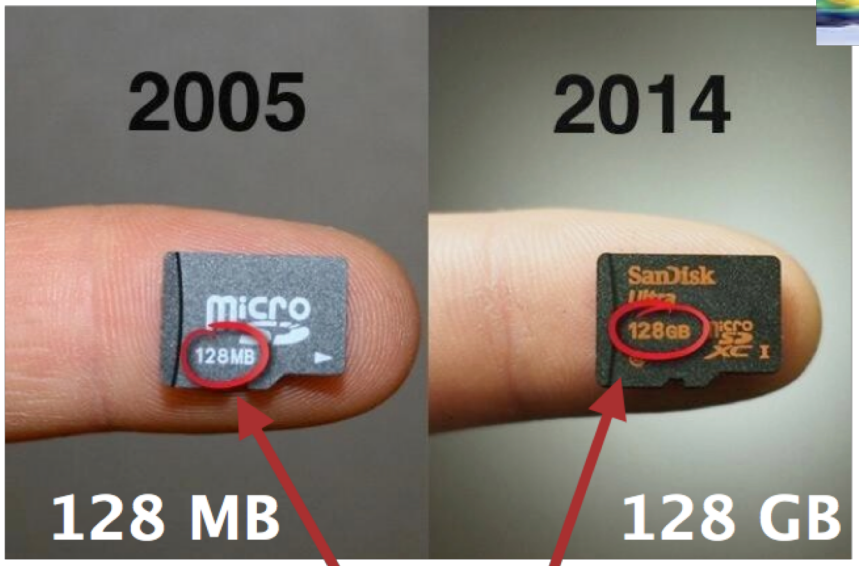
More real-world examples







Factor of 1000x increase in less than a decade!



Present day *real world*:

Phones: 100+ *Gigabytes*

Science and Business: 100s to 10,000s of *Petabytes*

Reasons behind Data Revolution

Rapid growth in computing capability has made data acquisition and generation much easier

- Esp. when compared with a much slower increase in I/O system bandwidth

High-resolution, multi-model scientific discovery requires and produces much more data

The needs that insights can be mined out of large amounts of low-entropy data have substantially increased over years

- Data-driven science v.s. model-driven (computational) science

Scientific breakthroughs are increasingly powered by advanced computing (HPC) plus data understanding capabilities

Our Vision

To create a holistic collection, management, and analysis software infrastructure of provenance data

- *Lightweight Provenance Service for high performance computing*

Objectives

- Run as an always-on service to collect and manage provenance for batch jobs transparently
- Capture comprehensive provenance with accurate causality to support a wide range of use cases, and
- Provide easy-to-use analysis tools for scientists and system administers to explore and utilize the provenance

What is Provenance

In general, provenance is documented history of an object and particularly useful to provide evidence for the originality of an art work

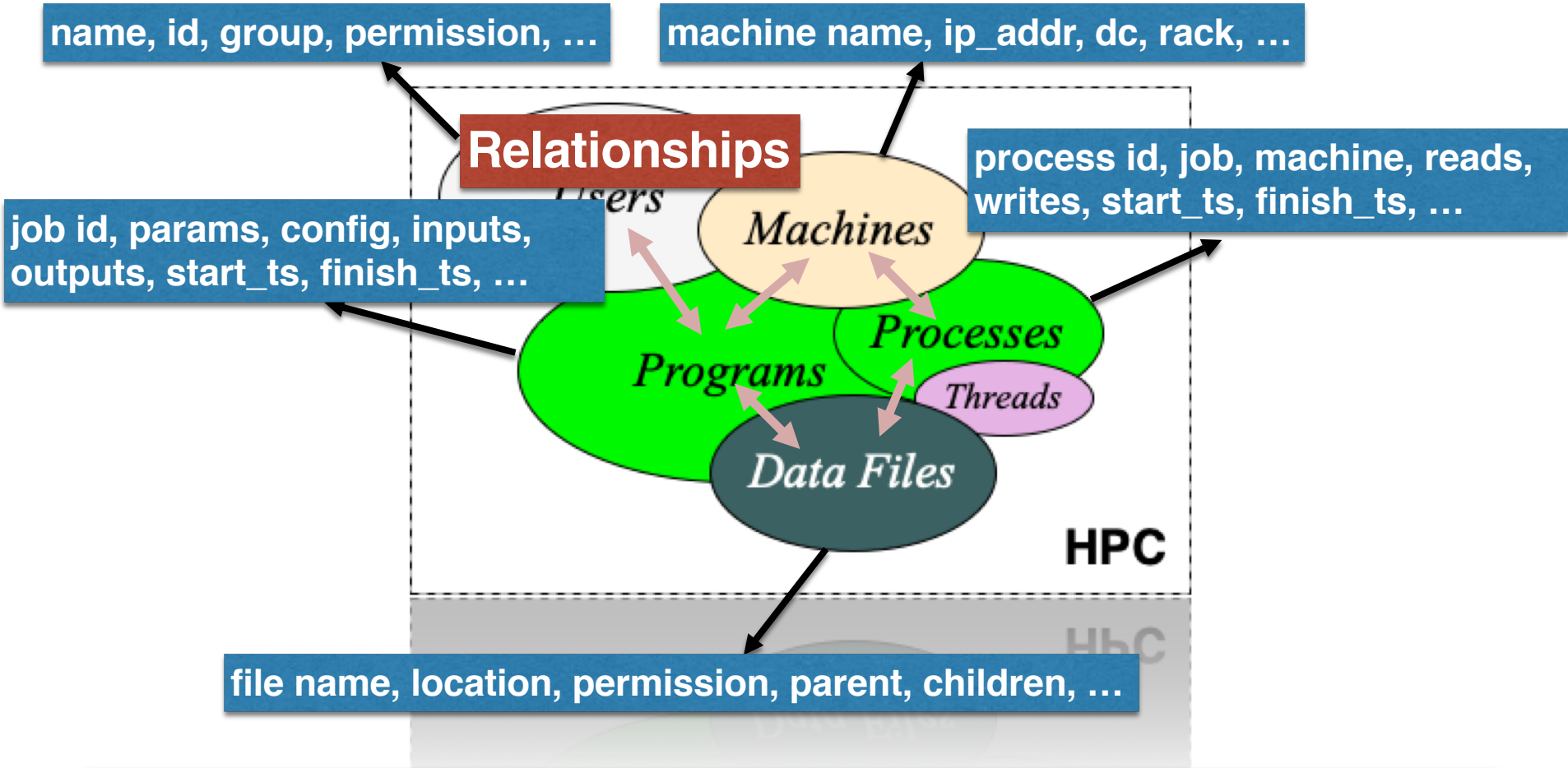
Little Dancer Aged Fourteen

1. Degas, Edgar (*created1878-1881*)
2. René De Gas (*heritage 1917*)
3. Adrien-Aurélien Hébrard (*a contract 5/3/1918)*
4. Nelly Hébrard (*heritage 1937)*
5. M. Knoedler & Company, Inc. *(cosigned 1955)*
6. Paul Mellon *(purchased 5/1956)*
7. National Gallery of Art (*bequest 1999*).

- From National Gallery of Art website

In computer science, provenance means the lineage of data, including processes that act on data and agents responsible for those processes.

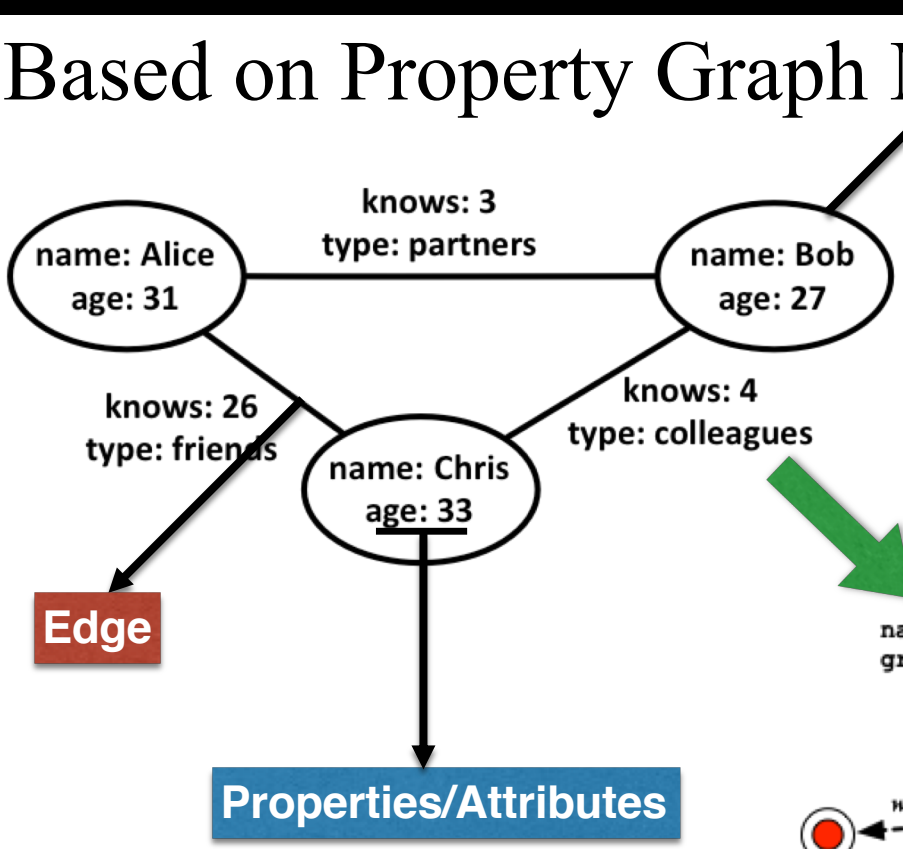
What is Provenance



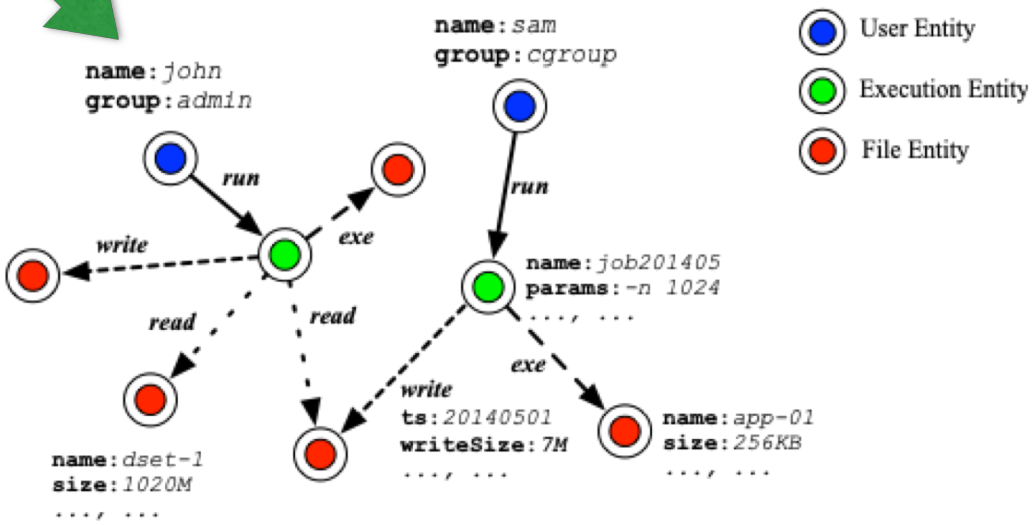
Provenance is data lineage from all entities and the relationships among all elements that contribute to the existence of the data.

How to Represent Provenance: A Graph-based Model for HPC Provenance Data

Based on Property Graph Model



Mapping HPC provenance onto a property graph model



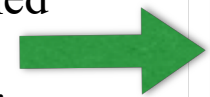
How to Represent Provenance: A Graph-based Model for HPC Provenance Data

Entity => Vertex

- Data Object: represents the basic data unit in storage
- Executions: represents applications including Jobs, Processes, Threads
- User: represents real end user of a system
- Allow to define your own entities

Relationship => Edge

- The relationship from Row to Column
- Reversed relationships also are defined
- *belongs/contains is general*
- Allow to define your own relationships



	User	Execution	Data Object
User		*run*	
Execution	*wasRunBy*	*belongs, contains*	*exe, read, write*
Data Object		*exedBy, wasReadBy, wasWrittenBy*	*belongs, contains*

Attributes => Property

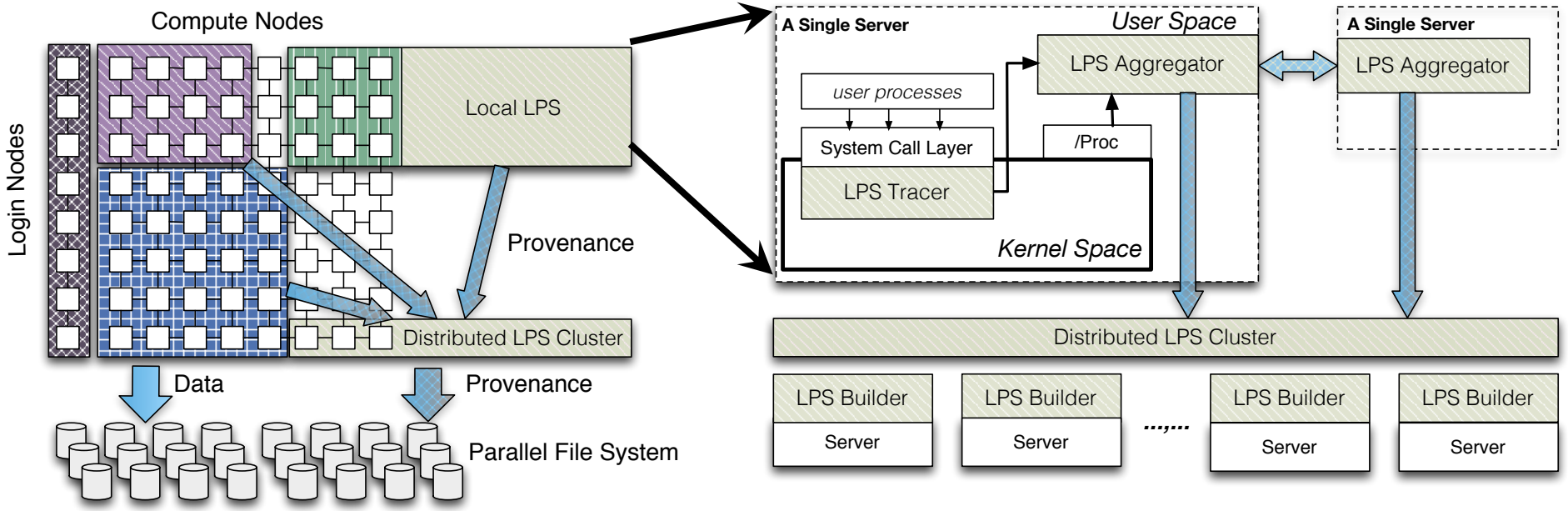
- Work on both Entity and Relationship
- Stored as Key-Value pairs attached on vertices and edges
- Allow to define your own properties

Requirements on Managing Provenance in HPC

- Performance Requirements:
  - HPC users are performance sensitive
  - Managing overhead should be less than 1% slowdown and less than 1MB memory footprint per core
- Coverage Requirements:
  - Provenance generated from multiple physical locations
  - Provenance could have various granularities
- Transparency Requirements:
  - Users should not change or recompile their codes for provenance
  - More aggressively, users should not disable it when provenance is used in critical missions

How to Collect and Manage Provenance

LPS Overall Architecture in HPC



How to Collect and Manage Provenance

LPS Tracer

LPS leverages kernel instrument to collect detailed runtime events to build provenance [among three methods]

Method	Transparency	No Privilege	Dynamic
Provenance APIs	×	✓	×
Library Wrapper	×	✓	×
Kernel Instrumentation	✓	×	✓

To support flexible granularity, it needs to enable/disable probing read/write events

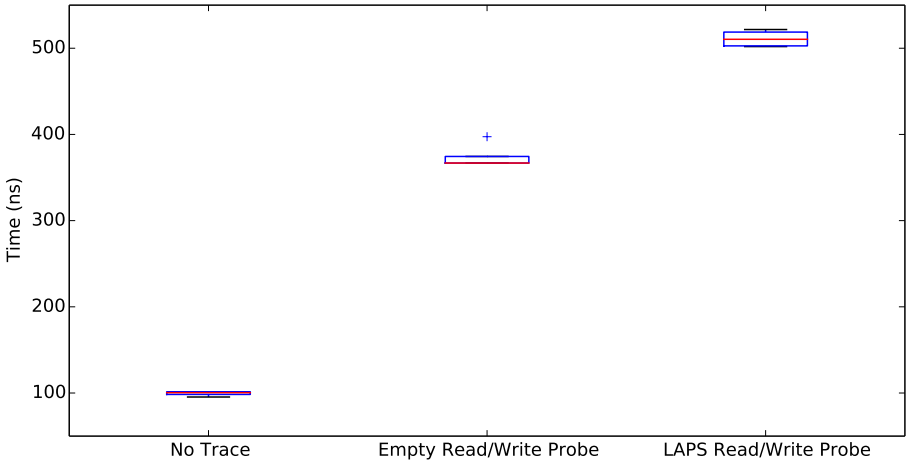
- Dynamic Probe
- Two kernel instrument scripts (Systemtap)
- The second one only probes read/write events
- Can be disabled/enabled accordingly in runtime

How to Collect and Manage Provenance

LPS Aggregator

1. Monitoring overhead and direct granularity change
2. Pruning noisy events to improve performance

- Instrumentation introduces overheads
  - Instrument Read/Write towards an application issuing 1M 1-byte writes
- The aggregator monitors read/write frequency
  - a counter records the events
  - a timer that resets the counter
  - notify and change granularity

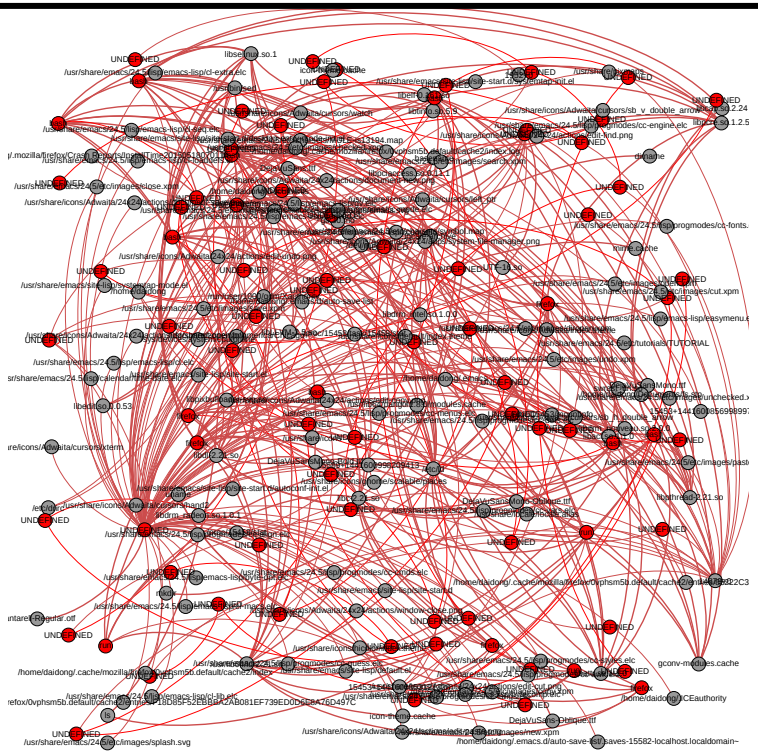


How to Collect and Manage Provenance

LPS Aggregator

1. Monitoring overhead and direct granularity change
2. Pruning noisy events to improve performance

Raw system events from kernel instrumentation

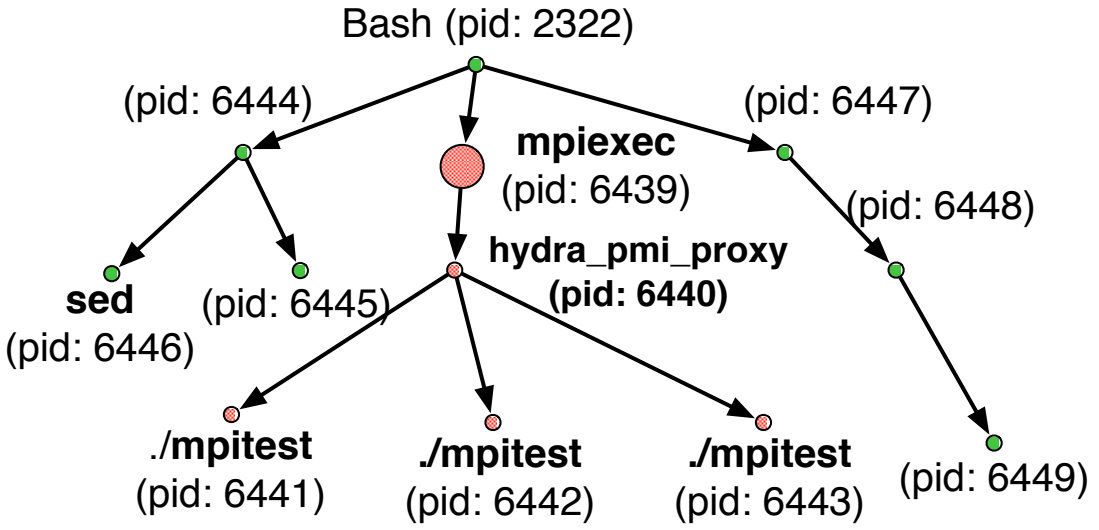


How to Collect and Manage Provenance

LPS Aggregator

1. Monitoring overhead and direct granularity change
2. Pruning noisy events to improve performance

- Representative Executions
  - Executions that users care the most
  - Eliminate unimportant child processes
  - Eliminate helper child processes
- Events of non-R executions are counted to their ancestor R executions



How to Collect and Manage Provenance

LPS Builder

- Local aggregators generate isolated provenance events
  - Workflows or jobs that are across multiple servers
- A global identifier challenge
  - To match identities in different machines needs a unique ID
  - Unique IDs are generated by specific software, no transparency
- A compromised solution
  - LPS relies on specific environmental variables to match identities
  - LPS should be notified about the name of these env variables
- Build provenance with versioning

What Can We Do with Provenance

HPC provenance is useful in the simulate-analyze-publish science discovery cycle

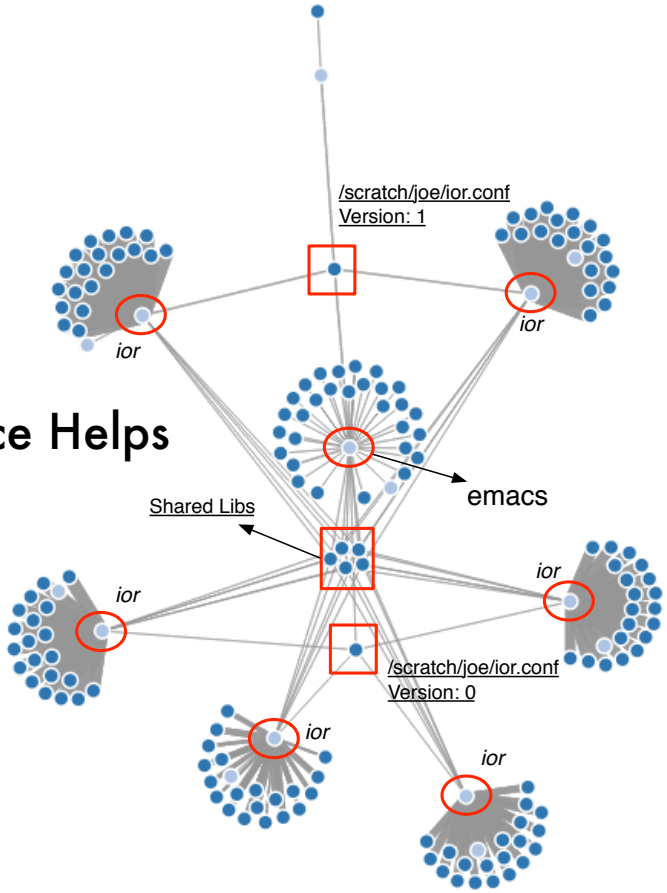
Evaluate a new system

- Repeatedly run the same benchmark **(typically time consuming)**
- Calculate avg and std for comparing

Questions

- If unexpected variations occur, how to ensure they are **from your system** or from **your evaluations**?
- Can others easily repeat the same evaluations?
- ...

Provenance Helps



What Can We Do with Provenance

Other use cases (not limited to)

User/project/job audit

Search -> Traversal -> Filter -> Traversal

- Provenance graph contains
  - *run relationships between Users and Executions*
  - *read/write relationships between Executions and Data Objects*
  - *additional attributes are also recorded with these relationships*

Organization of data space

- Present a logical layout of data sets to users
  - *In addition to traditional POSIX-style tree-structure directory*

Data sharing, publishing

Reproducibility, workflow management

What Can We Do with Provenance

The purpose of computing is insight, not numbers.

- Richard Hamming, 1962

Summary

- Data-driven discovery has become the new driving force for sciences, widely cited as the 4th paradigm
- We envision a holistic collection, management, and analysis software infrastructure of provenance data can be helpful for understanding data and mining insights
- We are working on a Lightweight Provenance Service infrastructure for HPC systems based upon prior R&D
- Call for more R&D efforts in this space and address challenges collectively from the community

Acknowledgements

- Prof. Dong Dai, UNCC
- Dr. Robert Ross, ANL
- Dr. Philip Carns, ANL
- Prof. William L. Hase, TTU
- Prof. Brian Ancell, TTU
- Prof. Alan Sill, TTU
- Ph.D. student: Mr. Misha Ahmadian

This project is supported in part by NSF Office of Advanced Cyberinfrastructure (OAC) through the Cyberinfrastructure for Sustained Scientific Innovation (CSSI) Program (Program Director: Dr. Vipin Chaudhary).

National Science Foundation
WHERE DISCOVERIES BEGIN

CAC
Cloud and Autonomic Computing Center

DISCL@TTU Team

Acknowledgements to our colleagues and collaborators

NSF CAC Cloud and Autonomic Computing Center

Thank You and Q&A

For more information please visit:

http://discl.cs.ttu.edu/

https://nsfcac.org/

yong.chen@ttu.edu

Cloud and Autonomic Computing Center

"Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Networking and Information Technology Research and Development Program."

The Networking and Information Technology Research and Development (NITRD) Program

Mailing Address: NCO/NITRD, 2415 Eisenhower Avenue, Alexandria, VA 22314

Physical Address: 490 L'Enfant Plaza SW, Suite 8001, Washington, DC 20024, USA Tel: 202-459-9674, Fax: 202-459-9673, Email: nco@nitrd.gov, Website: https://www.nitrd.gov