



RICE



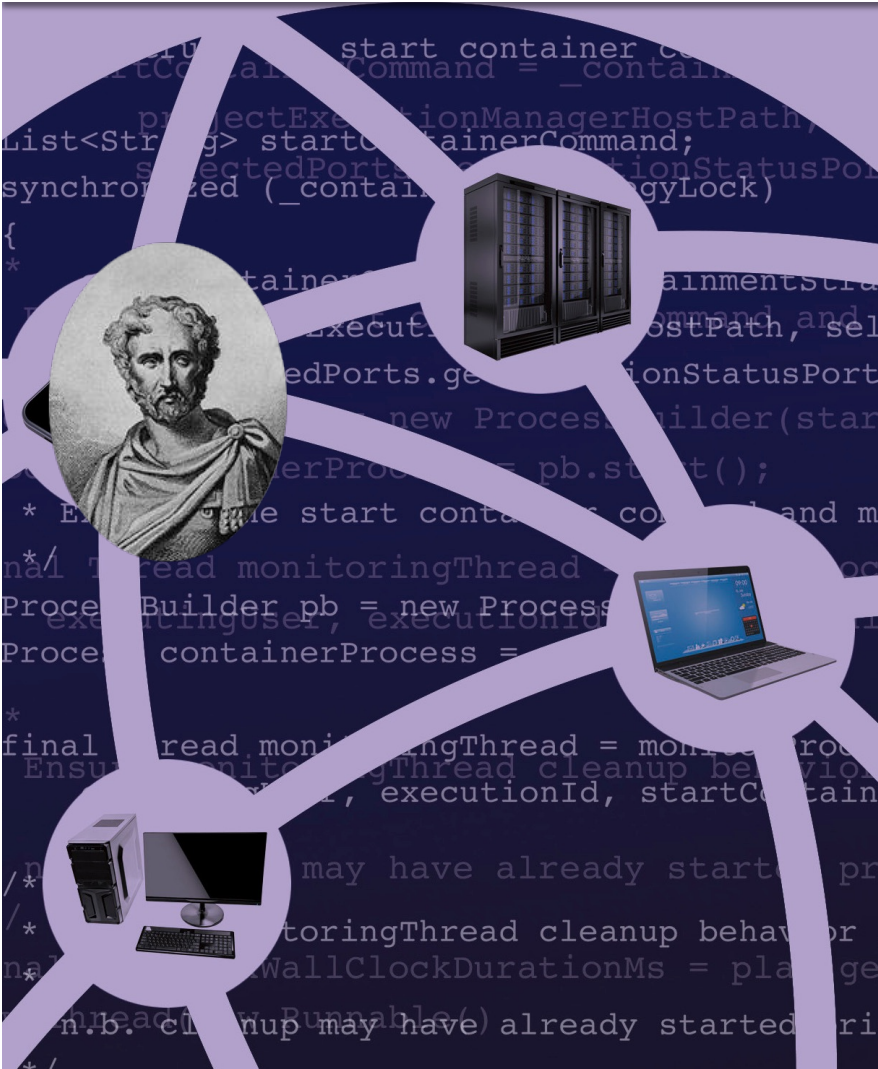
GRAMMATECH



WISCONSIN  
UNIVERSITY OF WISCONSIN-MADISON

# OPPORTUNITIES FOR SOFTWARE MINING AND ANALYTICS IN FUTURE HPC SOFTWARE DEVELOPMENT

Vivek Sarkar  
Rice University



03.25.2016



PLINY

SOS20 Workshop, 03/16

# Challenges for Exascale & Extreme Scale Systems

- Characteristics of Extreme Scale systems in the next decade
  - *Massively multi-core (~ 100's of cores/chip)*
  - *Performance driven by parallelism, constrained by energy & data movement*
  - *Subject to frequent faults and failures*
- Many Classes of Extreme Scale Systems



*Mobile, < 10 Watts,  
 $O(10^1)$  concurrency*



*Terascale Embedded,  
100's of Watts,  
 $O(10^3)$  concurrency*



*Petascale Departmental,  
100's of KW,  
 $O(10^6)$  concurrency*



*Exascale Data Center  
> 1 MW,  
 $O(10^9)$  concurrency*

## Key Challenges

- **Concurrency**
- **Energy efficiency**
- **Locality**
- **Resiliency**

## References:

- DARPA Exascale Study, 2008
- DARPA Exascale Software study, 2009



# Rice Habanero Extreme Scale Software Project

## Parallel Applications

### Unified execution model with semantic guarantees:

- 1) Lightweight asynchronous tasks and data transfers
  - Creation: *async tasks, future tasks, data-driven tasks*
  - Termination: *finish, future get, await*
  - Data Transfers: *asyncPut, asyncGet*
- 2) Locality control for task and data distribution
  - Computation and Data Distributions: *hierarchical places, global name space*
- 3) Inter-task synchronization operations
  - Mutual exclusion: *isolated, actors*
  - Collective and point-to-point operations: *phasers, accumulators*

**Habanero Programming Languages**  
(built on C/C++)

**Habanero Compiler & PIR**  
(Built on LLVM)

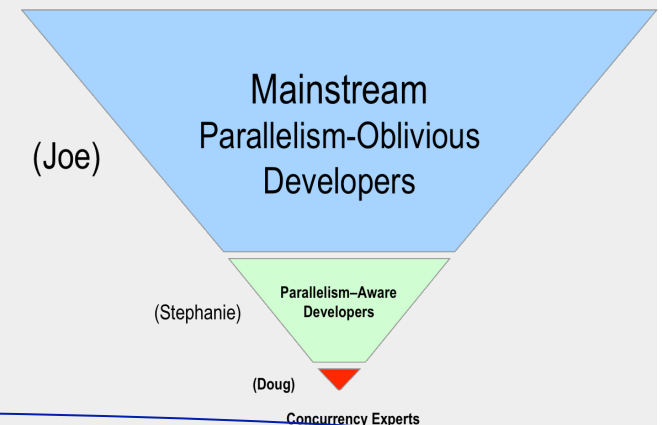
**Habanero Runtime System**  
(Built on MPI, GASNet, OCR, OpenSHMEM)

### Two-level programming model *Data Flow Graph Language for Domain Experts*

+

*Task-Parallel Languages & Libraries for Parallelism-aware Developers:*

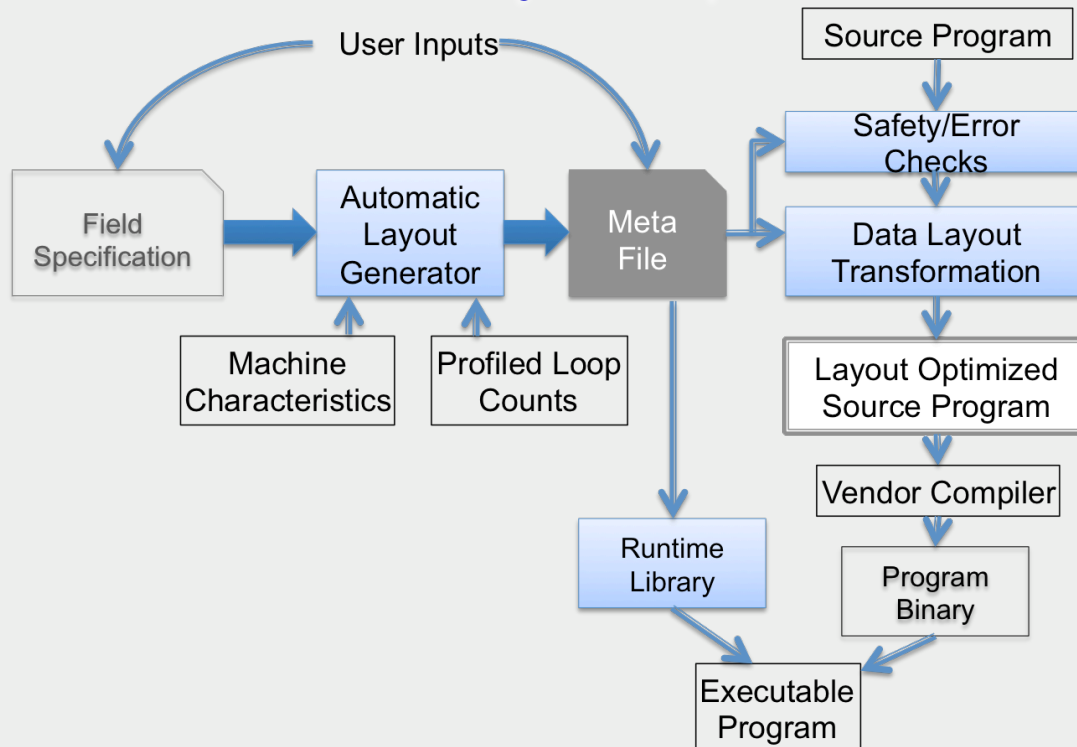
HC-MPI, Heterogeneous HC, OpenMP 4.5, Habanero-C++, Habanero-UPC++, Asynchronous OpenSHMEM, ...



## Extreme Scale Platforms



# Automatic data layout optimization using the TALC tool



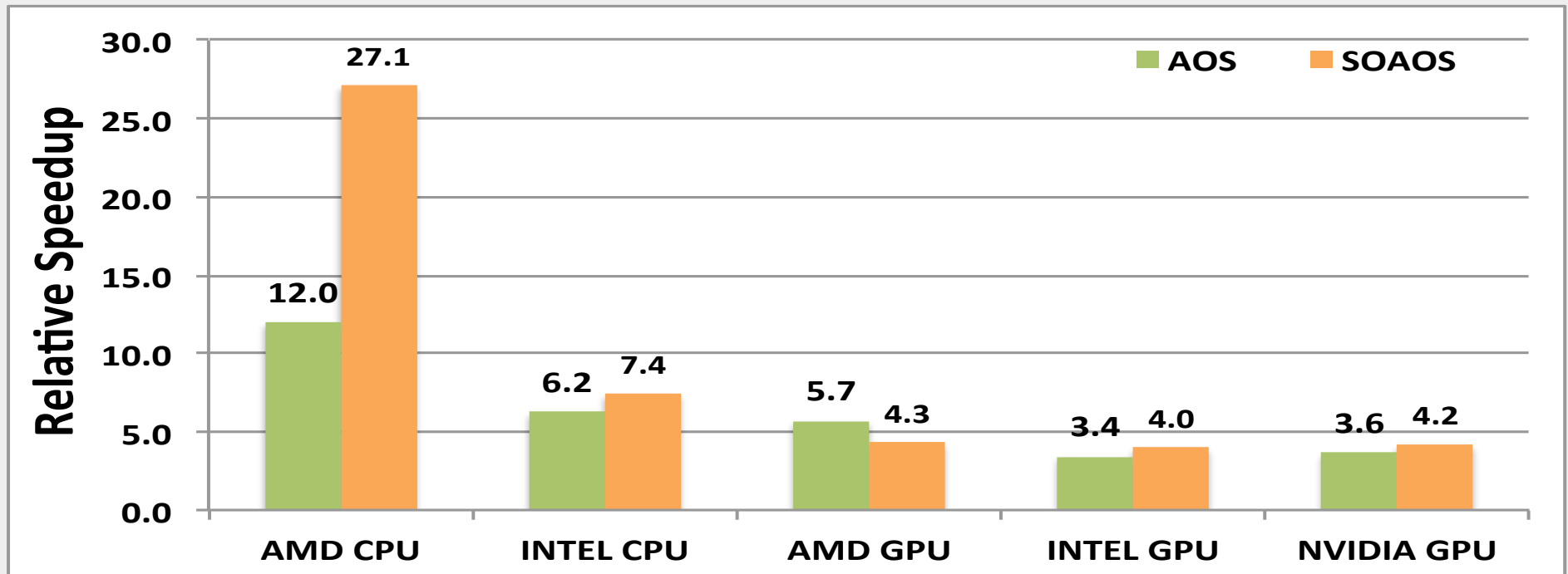
```

#pragma omp parallel for private(jj,ii,i)
for (kk = kmin; kk < kmax; kk++) {
  for (jj = jmin; jj < jmax; jj++) {
    for (ii = imin; ii < imax; ii++) {
      i = ii + jj * jp + kk * kp;
      b[i] = dbl[i] * xdbl[i] + dbc[i] * xdbc[i] + dbr[i] * xdbr[i] +
        dcl[i] * xdcl[i] + dcc[i] * xdcc[i] + dcr[i] * xdcr[i] +
        dfl[i] * xdfl[i] + dfc[i] * xdfc[i] + dfr[i] * xdfr[i] +
        cbl[i] * xcbl[i] + cbc[i] * xcbc[i] + cbr[i] * xcbr[i] +
        ccl[i] * xccl[i] + ccc[i] * xccc[i] + ccr[i] * xccr[i] +
        cfl[i] * xcfl[i] + cfc[i] * xcfc[i] + cfr[i] * xcfr[i] +
        ubl[i] * xubl[i] + ubc[i] * xubc[i] + ubr[i] * xubr[i] +
        ucl[i] * xucl[i] + ucc[i] * xucc[i] + ucr[i] * xucr[i] +
        ufl[i] * xufl[i] + ufc[i] * xufc[i] + ufr[i] * xufr[i];
    }
  }
}
    
```

IRSmk performance relative to default 27x1 layout (bigger is better)

Platform	27 × 1	9 × 3	3 × 9	1 × 27
IBM POWER7	1.00	4.66	4.66	4.71
AMD APU	1.00	1.26	1.38	1.40
Intel Sandybridge	1.00	1.06	1.10	1.10
IBM BG/Q	1.00	1.65	2.14	2.20

## Data Layout Transformations are important for GPUs too ...

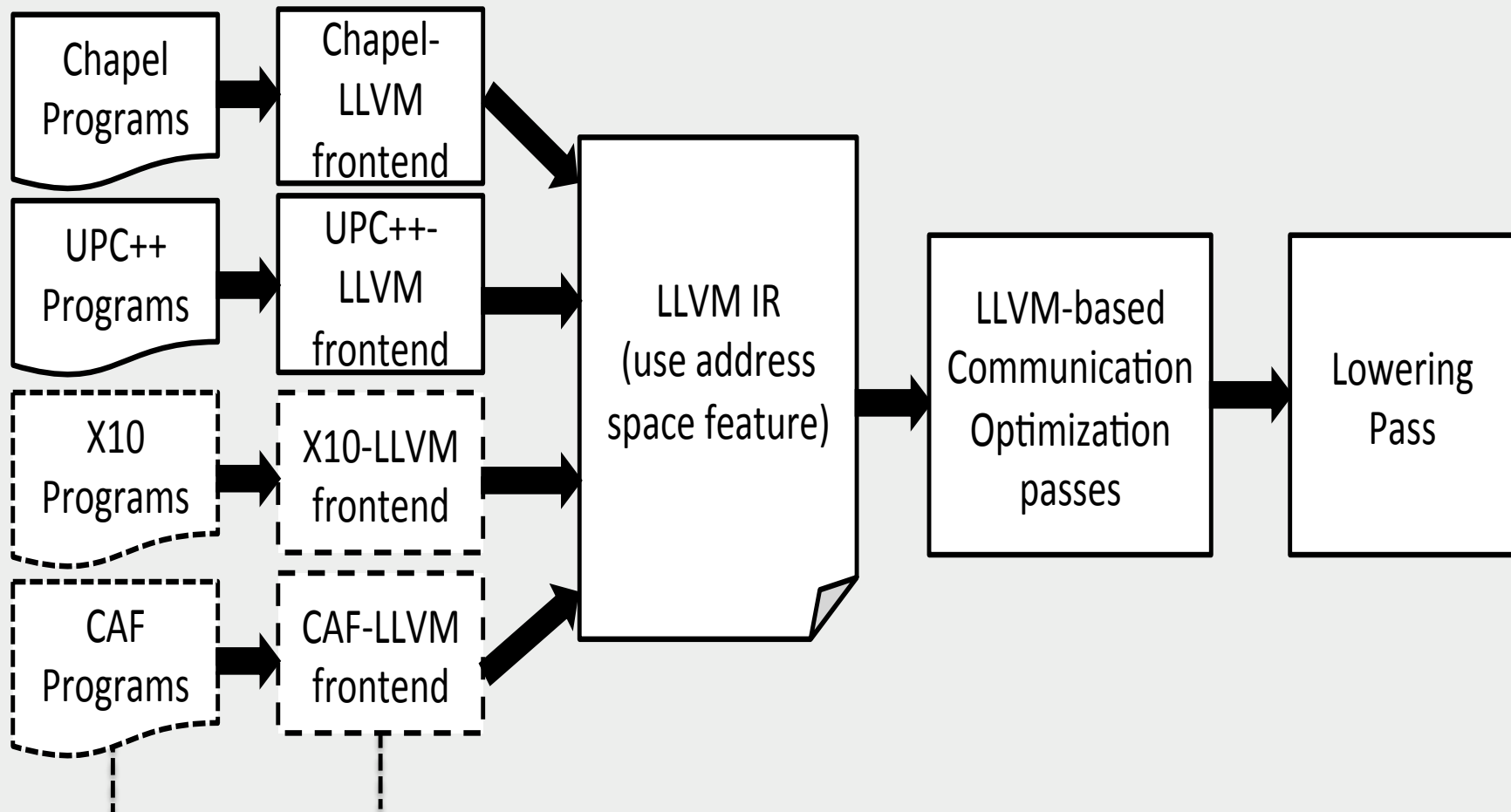


Speedup Relative to default SoA Layout

- CPUs benefit from spatial locality
  - AoS (Array of Structures)
- GPUs benefit from coalescing
  - SoA (Structure of Arrays)

“Automatic Data Layout Generation and Kernel Mapping for CPU+GPU Architectures”. Deepak Majeti, Kuldeep Meel, Raj Barik and Vivek Sarkar. 25th International Conference on Compiler Construction (CC 2016), March 2016.

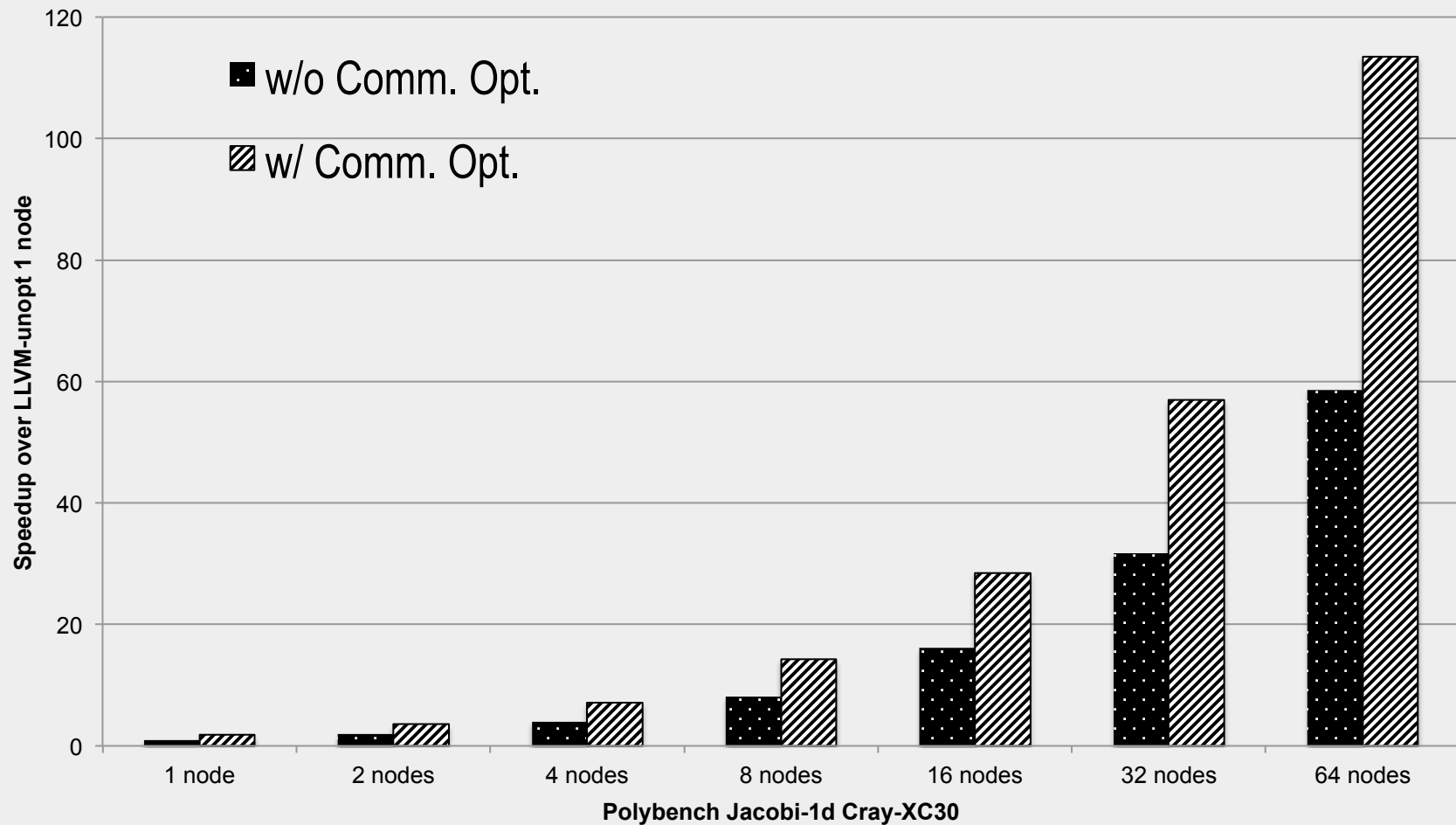
# Common LLVM-based Communication Optimization Framework for Multiple Languages



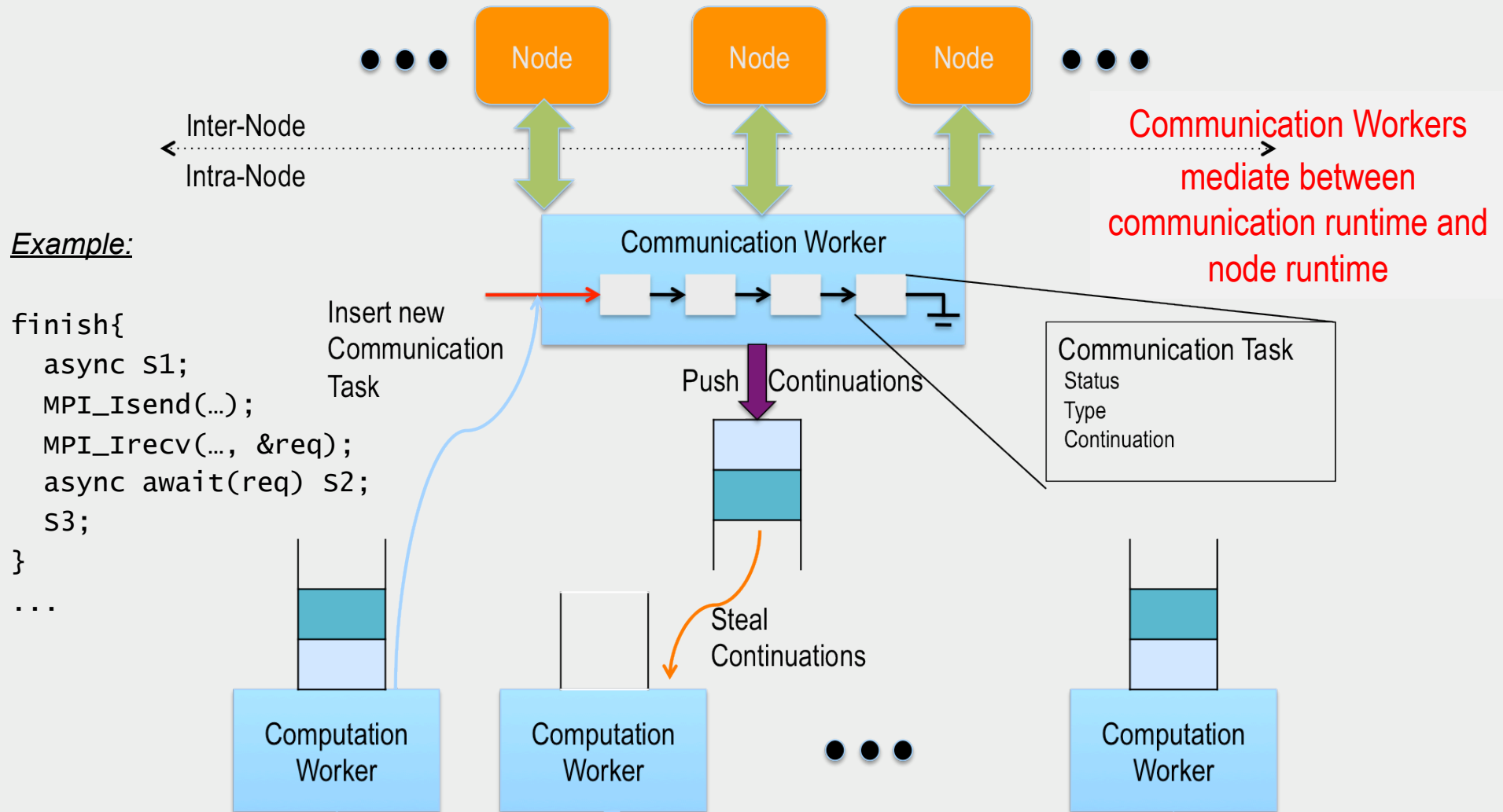
“LLVM-based Communication Optimizations for PGAS Programs,” Akihiro Hayashi, Jisheng Zhao, Michael Ferguson, Vivek Sarkar.



# Performance improvement due to Communication Optimization for Jacobi example in UPC++



# Integrating Inter-node Communication with Intra-node Task Scheduling

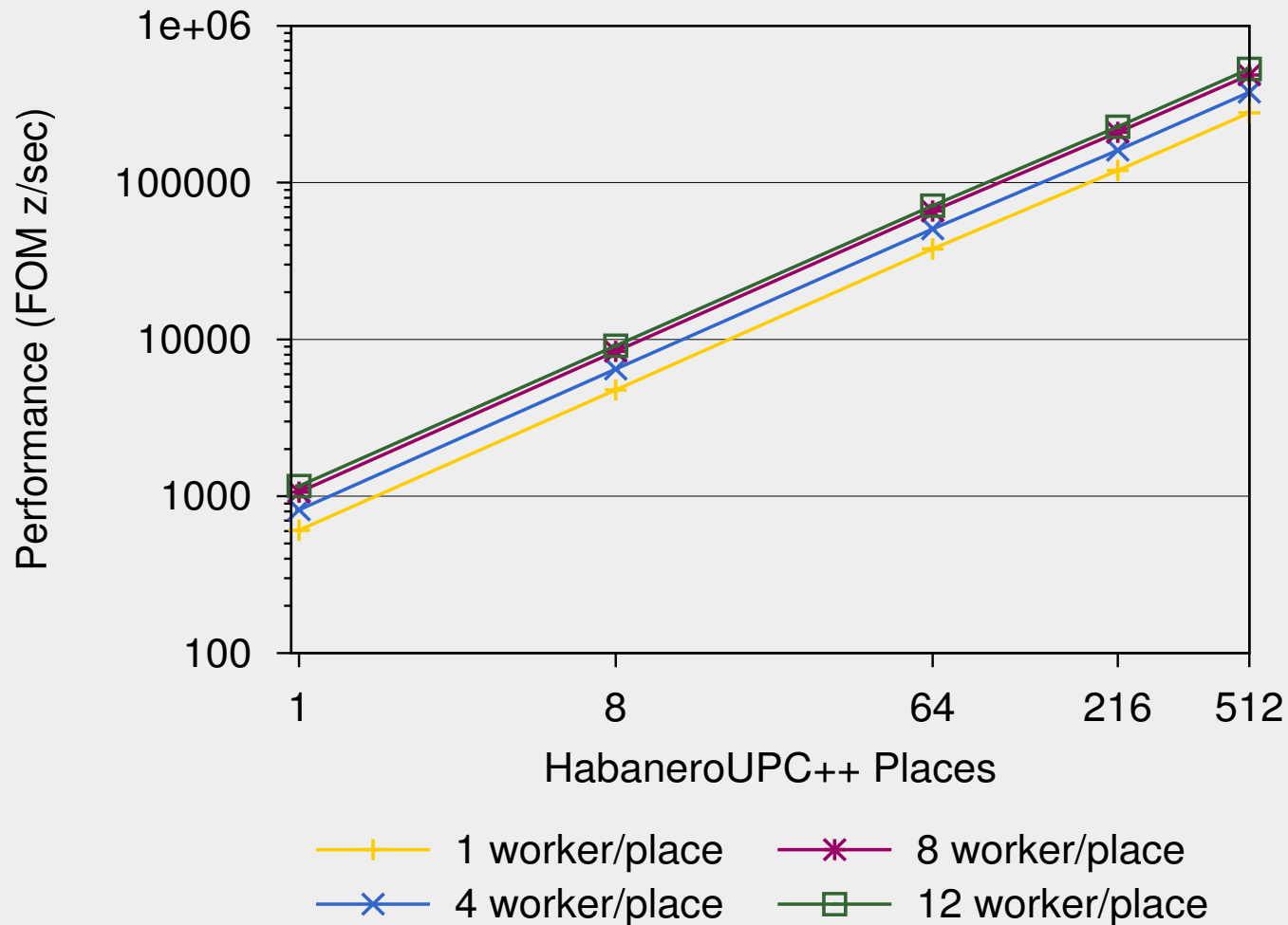


"Integrating Asynchronous Task Parallelism with MPI." Sanjay Chatterjee, Sagnak Tasirlar, Zoran Budimlic, Vincent Cave, Milind Chabbi, Max Grossman, Yonghong Yan, Vivek Sarkar. IPDPS 2013.





# Weak Scaling Result for Habanero-UPC++ version of LULESH on NERSC Edison system



“HabaneroUPC++: A Compiler-free PGAS Library.” Vivek Kumar, Yili Zheng, Vincent Cavé, Zoran Budimlić, Vivek Sarkar. PGAS 2014.



# Open Community Runtimes (OCR) Building Blocks

- Data Blocks (DBs)
  - contains semantically-meaningful metadata that runtime can use
  - relocatable by runtime for power, reliability, ...
    - accessed via globally unique ids (GUIDs)
  - allows exploitation of heterogeneous memories (NUMA, scratchpads, ...)
- Event-driven tasks (EDTs)
  - Can be processes, threads, functions, codelets...
  - An EDT can contain internal data parallelism  

```
u8 ocrEdtCreate(ocrGuid_t * guid, ocrGuid_t templateGuid, u32 paramc, u64* paramv,  
u32 depc, ocrGuid_t *depv, u16 properties, ocrGuid_t affinity, ocrGuid_t *outputEvent);
```
- Events (Dependences)
  - specified explicitly as contingencies on which EDTs are initiated
  - several types of dependences
  - dependences are specified as GUIDs throughout the system



# Can we improve productivity by capturing “institutional knowledge” related to performance portability?

## Sources of heterogeneity:

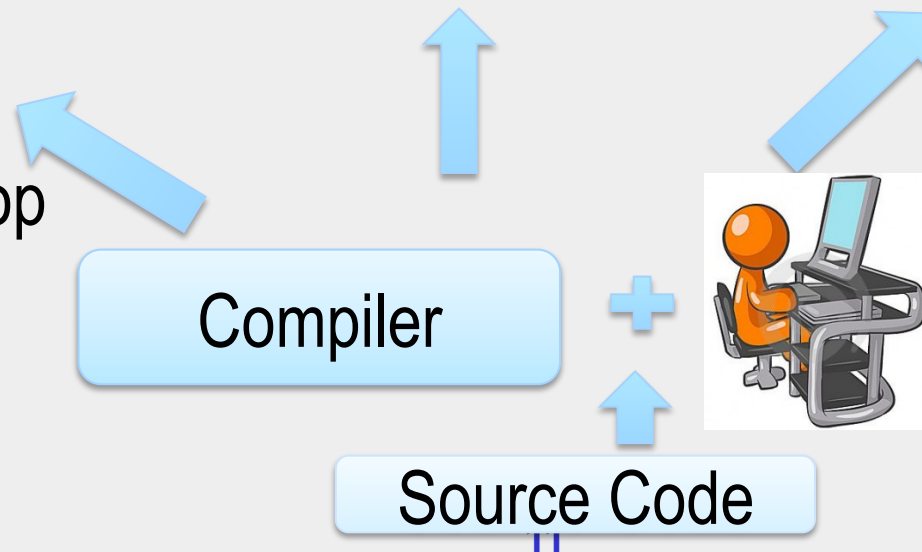
- Processors
- Memory
- Interconnect
- . . .



Desktop/Laptop Processors

Supercomputers

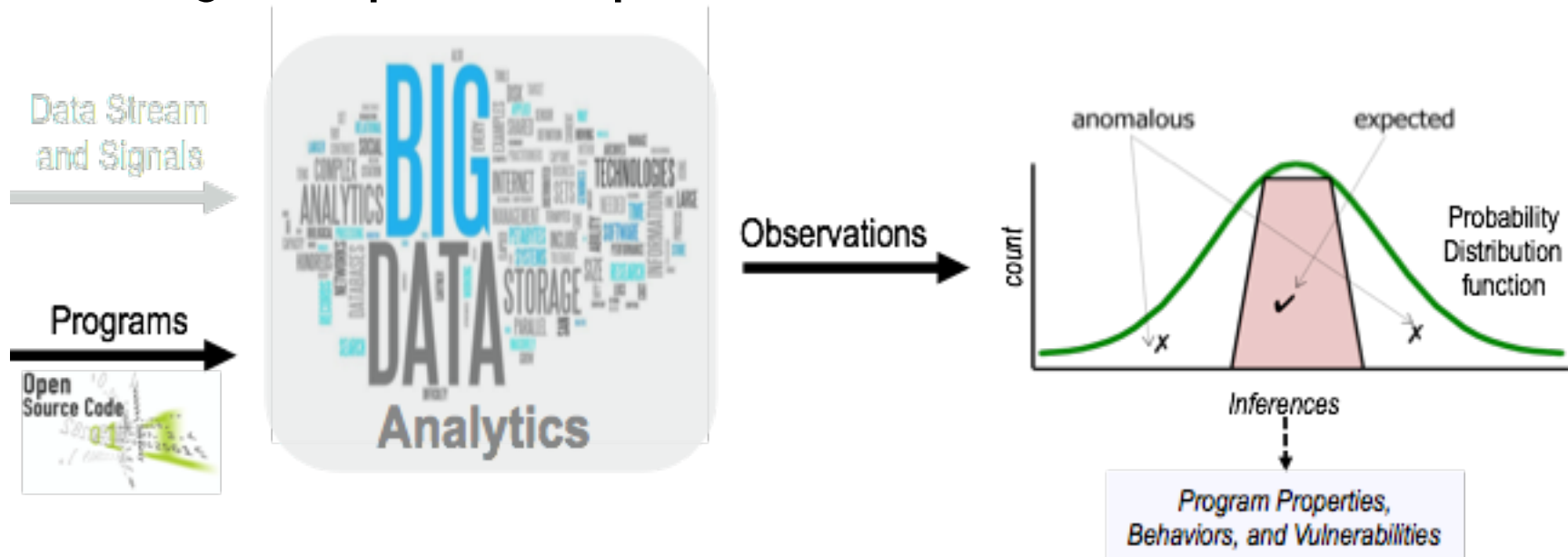
Server Systems



*Programmers spend significant amount of time tuning performance for different platforms*

# DARPA program on Mining and Understanding Software Enclaves (MUSE)

- Goal: Apply principles of Big Data Analytics to a large corpus of Open-Source Software



- Core idea: treat programs (semantic objects extracted from programs) as data
- Source: [http://www.darpa.mil/Our\\_Work/I2O/Programs/Mining\\_and\\_Understanding\\_Software\\_Enclaves\\_\(MUSE\).aspx](http://www.darpa.mil/Our_Work/I2O/Programs/Mining_and_Understanding_Software_Enclaves_(MUSE).aspx)



RICE



GRAMMATECH

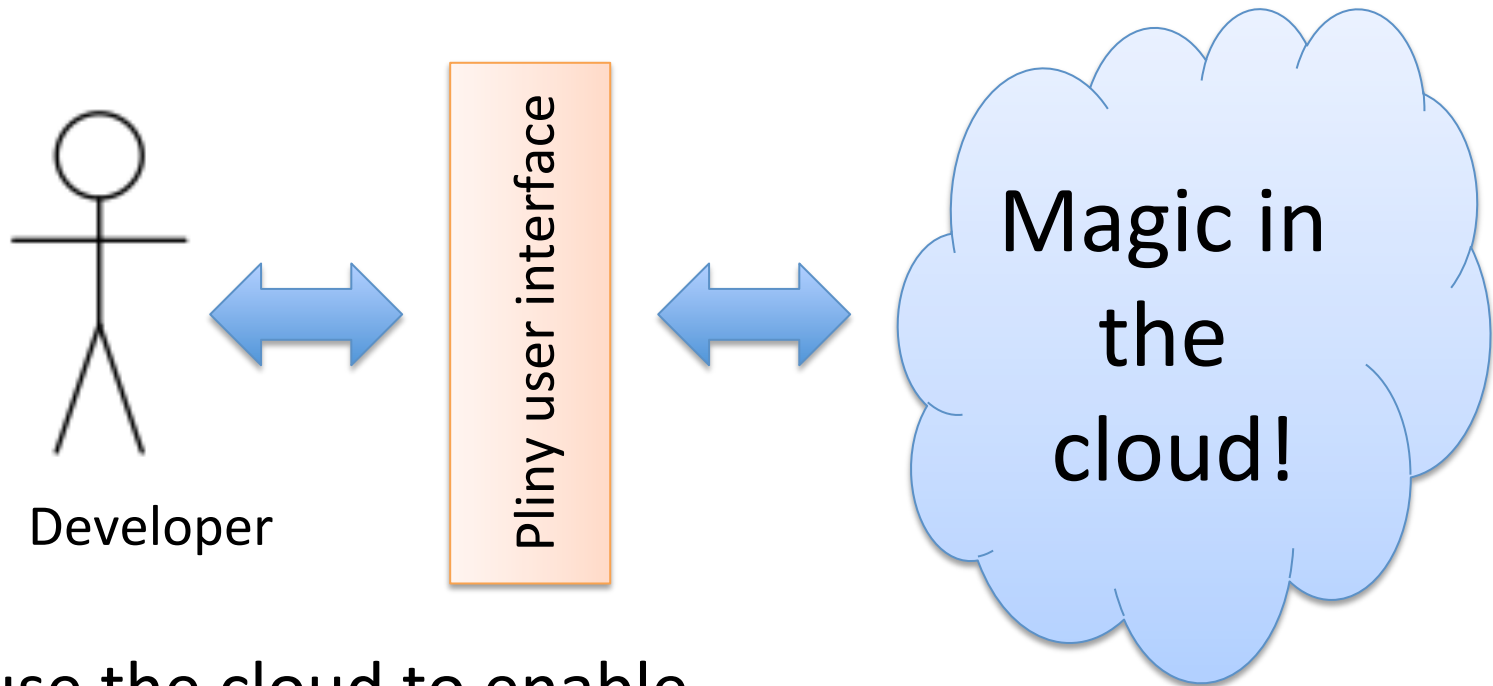


WISCONSIN  
UNIVERSITY OF WISCONSIN-MADISON

# Pliny Team

- **Rice:** Vivek Sarkar (PI), Swarat Chaudhuri, Chris Jermaine (Faculty), Michael Burke, Philippe Charles, Carlos Monroy, Kia Teymourian, Jisheng Zhao (Research Scientists), Tiago Cogumbreiro, Hassan Eldib, Vijayaraghavan Murali (Postdoctoral Researchers), John Feser, Yanxin Lu, Afsaneh Rahbar, Rishi Surendran (PhD Students)
- **Grammatech:** David Melski (VP of Research), Denis Gopan, Vineeth Kashyap (Senior Scientists), Duc Nguyen, Anurag Singh (Software Engineer)
- **UT Austin:** Isil Dillig, Thomas Dillig (Faculty), Ruben Martins (Postdoctoral Researcher), Yu Feng, Arati Kaushik, Yuepeng Wang, Navid Yaghmazadeh (PhD Students)
- **UW Madison:** Ben Liblit, Thomas Reps (Faculty), Jason Breck, David Bingham Brown (PhD Students)

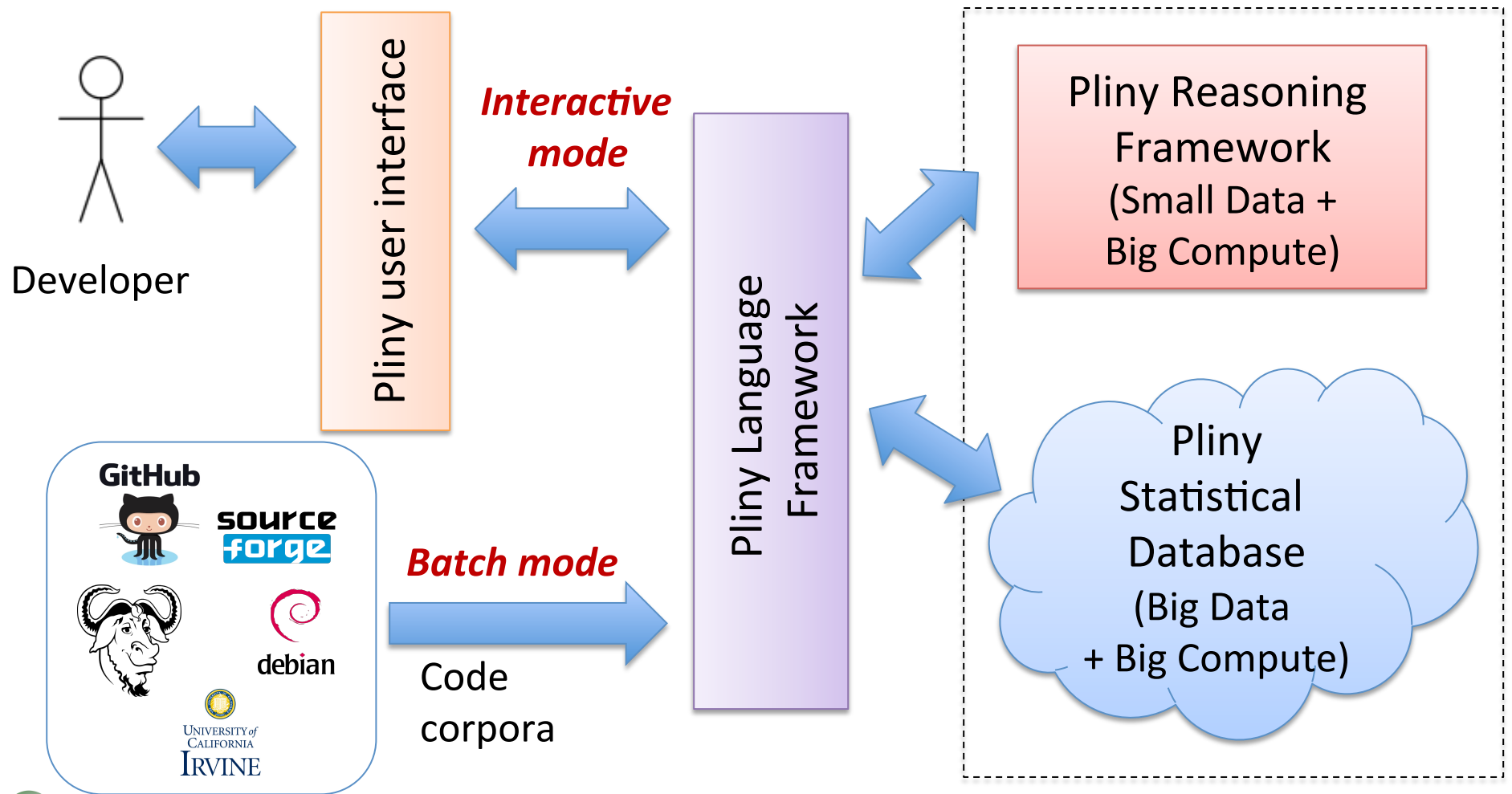
# The Pliny vision



Goal: use the cloud to enable order-of-magnitude improvements in developer productivity and software quality

- Debugging
- Repair
- Program Synthesis

# Using Big Code to Realize the Pliny Vision



# Early Publicity on Wired.com

## The \$11M Tool That Could Help Computers Write Their Own Code

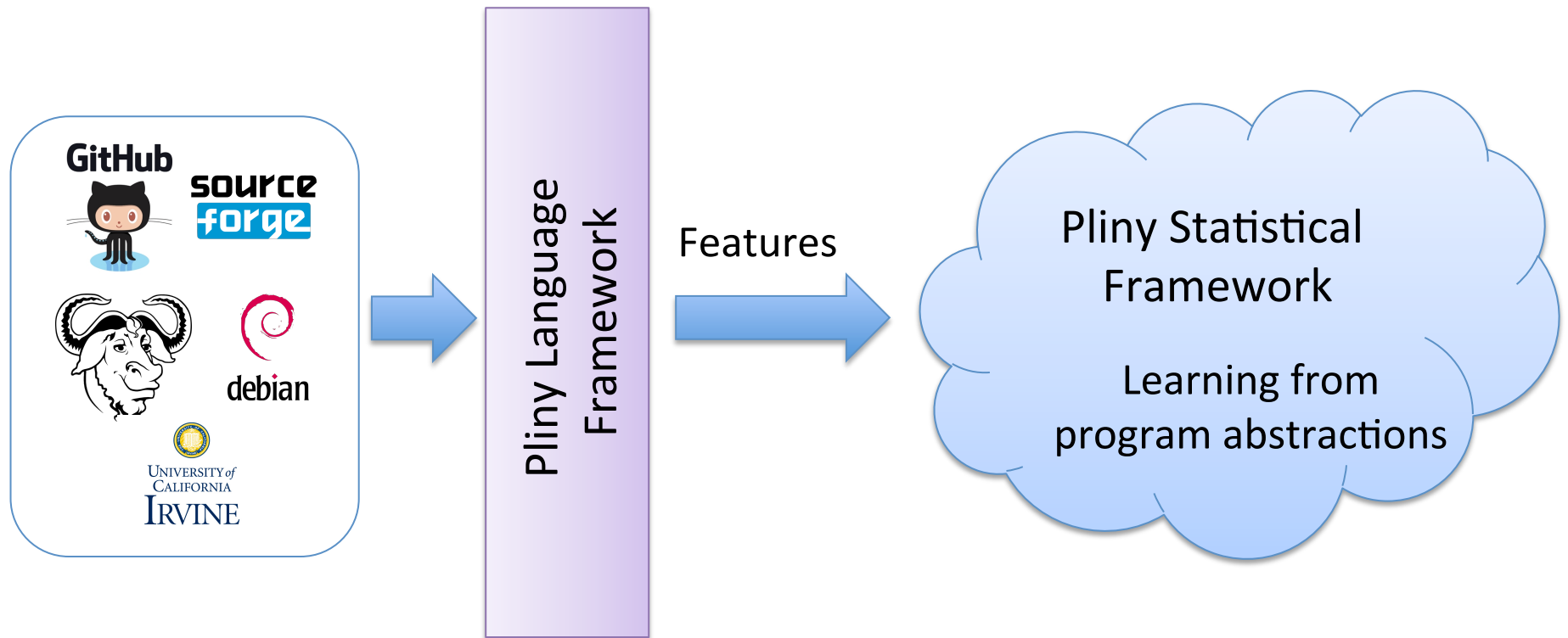
BY KLINT FINLEY 11.07.14 | 6:30 AM | PERMALINK



- "... The PLINY team will begin by analyzing open source code from around the web, drawing on code hosting services like GitHub and Sourceforge, along with various major open source projects, such as those managed by the Apache Foundation. Eventually, though, he envisions a corporate version that will index all of a company's own proprietary software projects. The team is also building a custom database system specifically designed for the purpose of storing and analyzing code. The new database will give them ways to structure and prioritize the code it indexes. This could help with the code quality issue. Projects known for exceptionally good good could be prioritized, or perhaps code written by specific people would be given preference. The end result could be something that looks an awful like Google's autocomplete—only more useful."
- Source: <http://www.wired.com/2014/11/darpa-pliny/>



# The Pliny approach: Batch mode

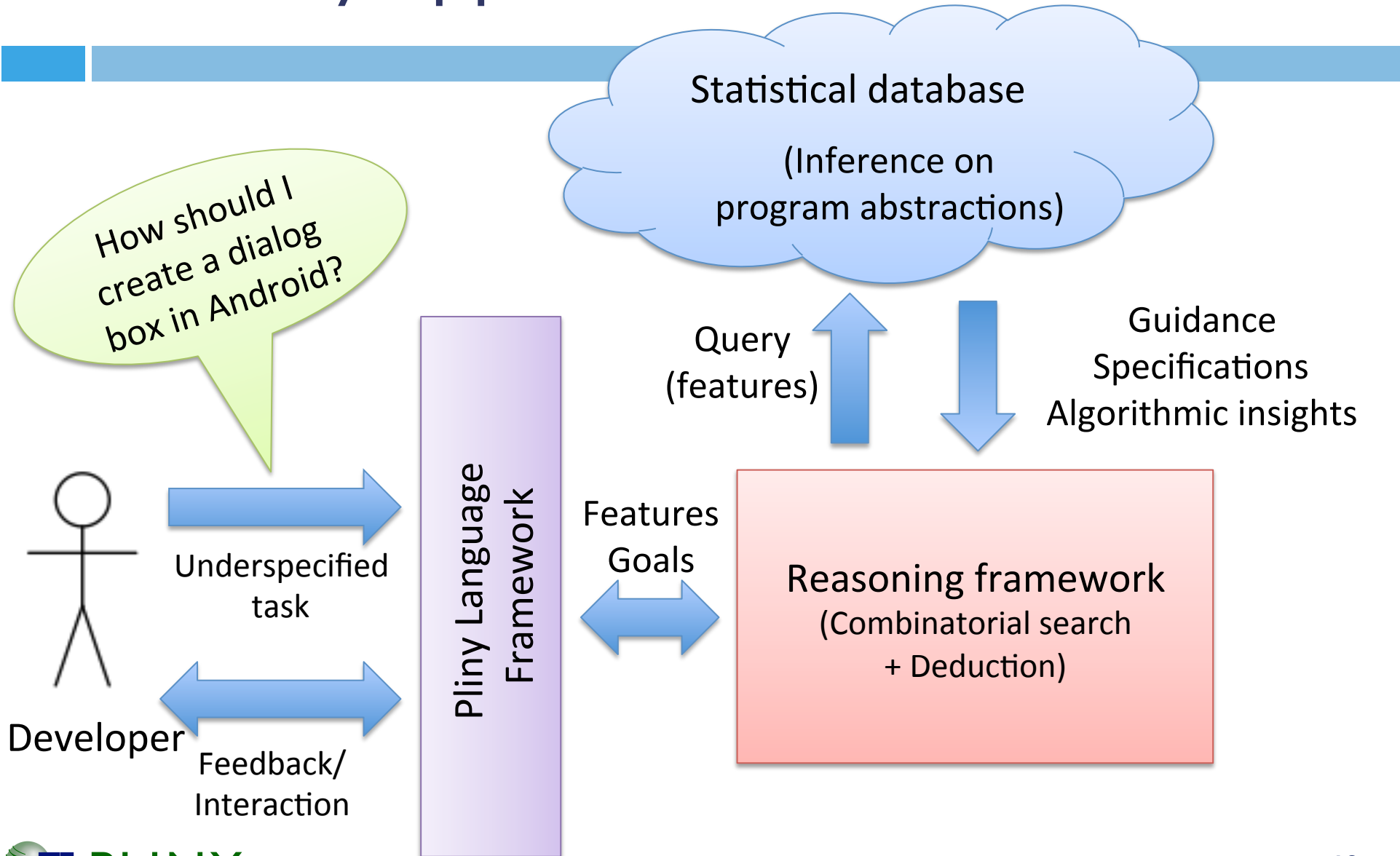


# The Pliny Statistical Framework

Large-scale machine learning on program features

- k-nearest neighbors
- Markov random fields
- HMMs and generalizations
- ...

# The Pliny approach: Interactive mode



# The Pliny reasoning framework

Functional synthesis  
problem



Pliny reasoning  
framework



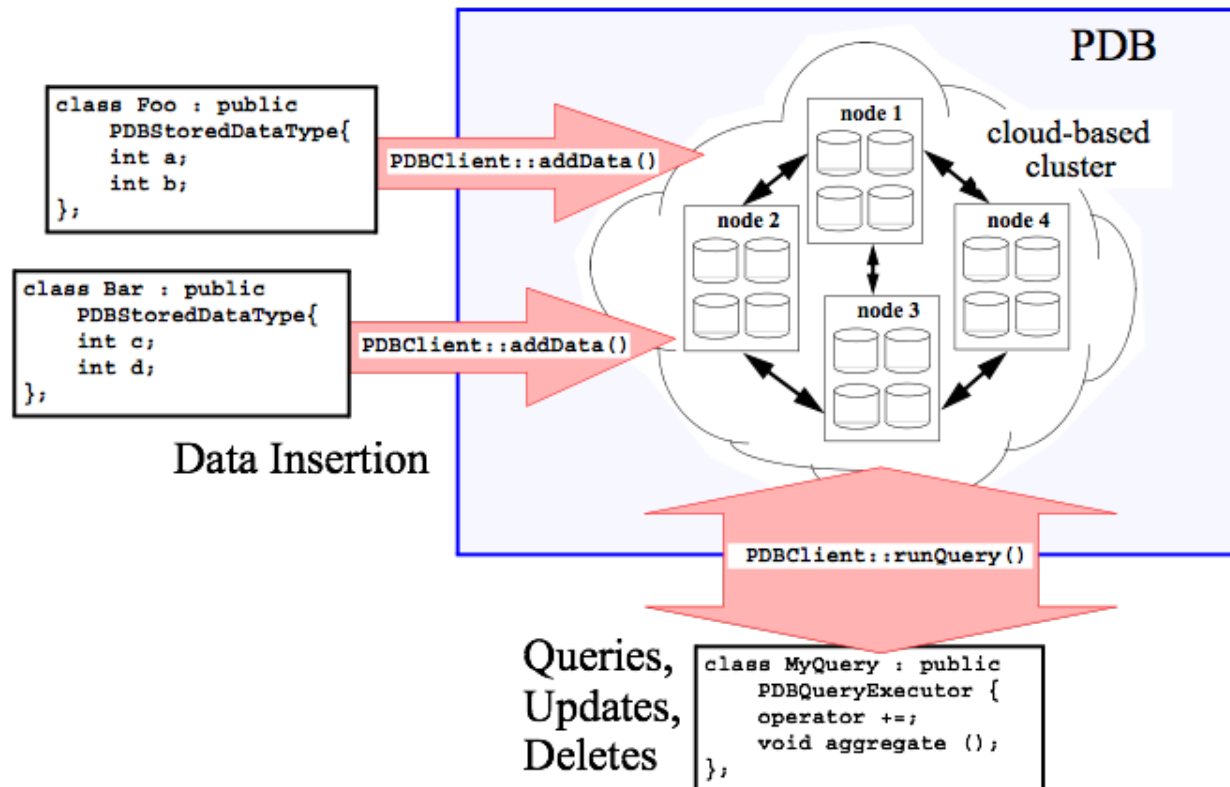
Solution to  
synthesis instance

**Find a mathematical function that...**

1. ... satisfies a set of logical constraints
2. ... can be expressed in a syntactic space built from corpus components
3. ... is optimal by a set of quantitative criteria

**Combinatorial search**  
+  
**Automated deduction**

# The Pliny Database and Compute Engine (PDB)

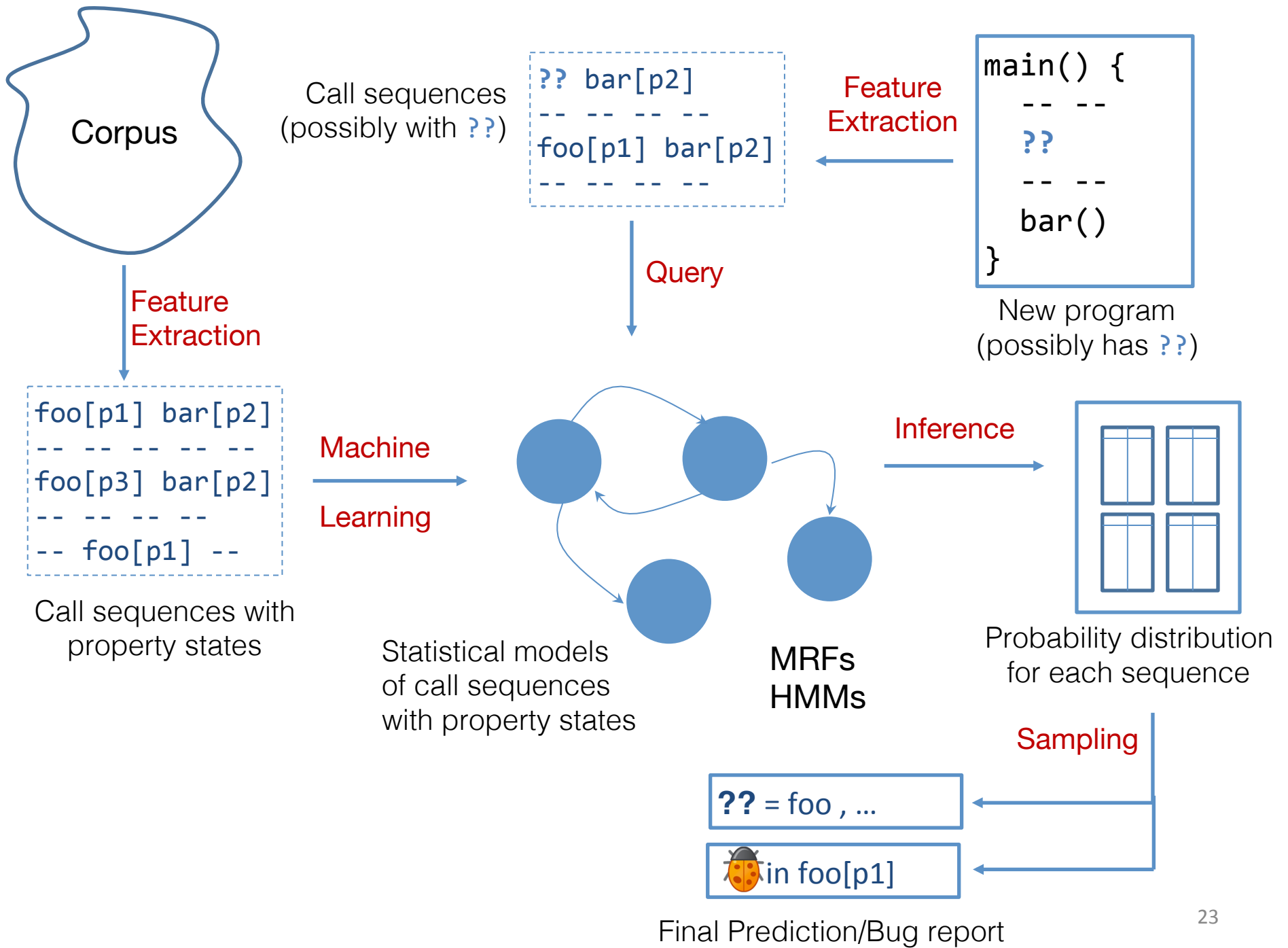


- ❖ Flexible object model, no distinction between RAM/network/secondary storage layout




# Example: Learning API specifications

- Android: “Dialog boxes typically contain at least one button or a message”
- POSIX: “Always read only from a file that has been opened”

**Learn from sequences of calls,  
and constraints among their arguments,  
generated from real code**



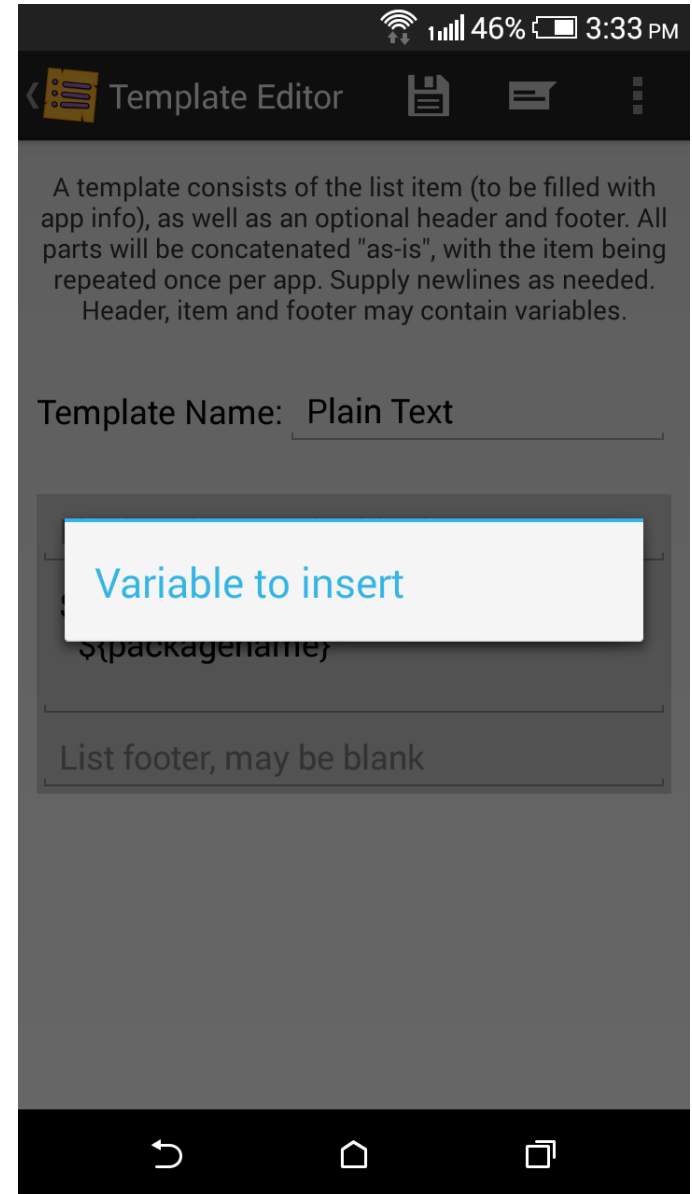
# Learning API specifications

Corpus	Programming Language	Number of packages	Lines of code	Size of corpus	Source
 <b>debian</b>	C	3500	256 million (preprocess)	200 GB (source+ compiled)	<a href="http://www.debian.org">www.debian.org</a> <a href="http://wiki.debian.org/Debtags">wiki.debian.org/Debtags</a>
 UNIVERSITY of CALIFORNIA <b>IRVINE</b> Sourcerer	Java	74,000	630 million	433 GB (source+jar)	<a href="http://sourcerer.ics.uci.edu">sourcerer.ics.uci.edu</a>
 <b>ANDROID</b>	Java	2500	N/A (APK bytecode)	28 GB (APK only)	<a href="http://www.androiddrawer.com">www.androiddrawer.com</a> <a href="http://www.fdroid.org">www.fdroid.org</a>



# Learning API specifications (Rice)

- Android Dialog box API
  - 2500 packages, 75,000 sequences
  - Training time: 6 hours on 3.2GHz x 20 cores
- Finding UI bugs in the wild
  - Example: Google Play Store app “List My Apps” (50k downloads, 4.3/5 stars) found to violate Dialog box API spec
  - Displayed dialog box without any content to select
  - Inference time: 5-10 secs



# Pliny's Open Architecture

PDB's open API can be used by different feature extractors for different languages and different similarity metrics

Feature Extraction  
(Small Data + Big Compute)

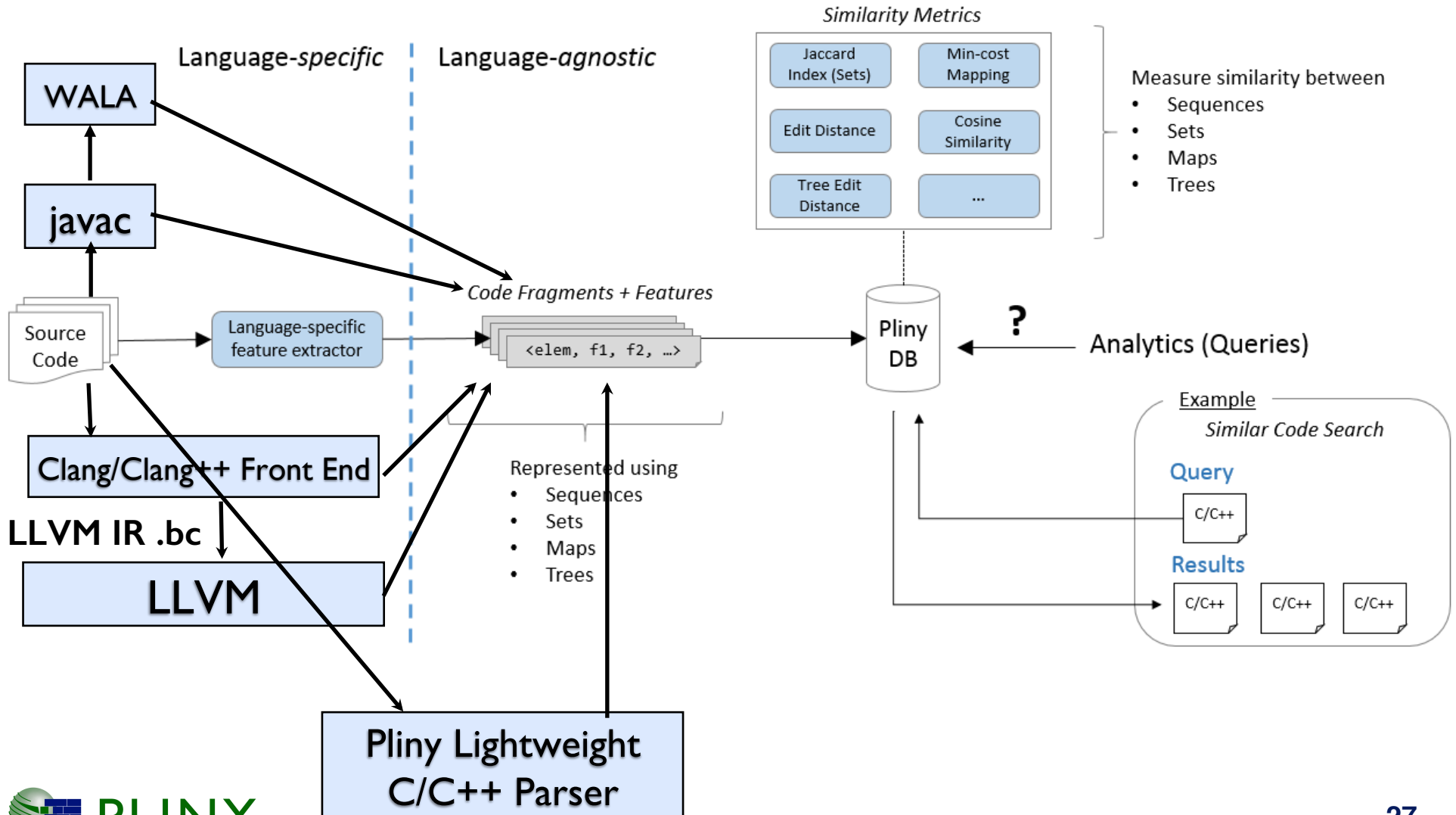
PDB:  
24/7 Pliny  
Statistical  
Database &  
Analytics  
Platform  
(Big Data + Big Compute)

Pliny Reasoning Framework  
(Small Data + Big Compute)

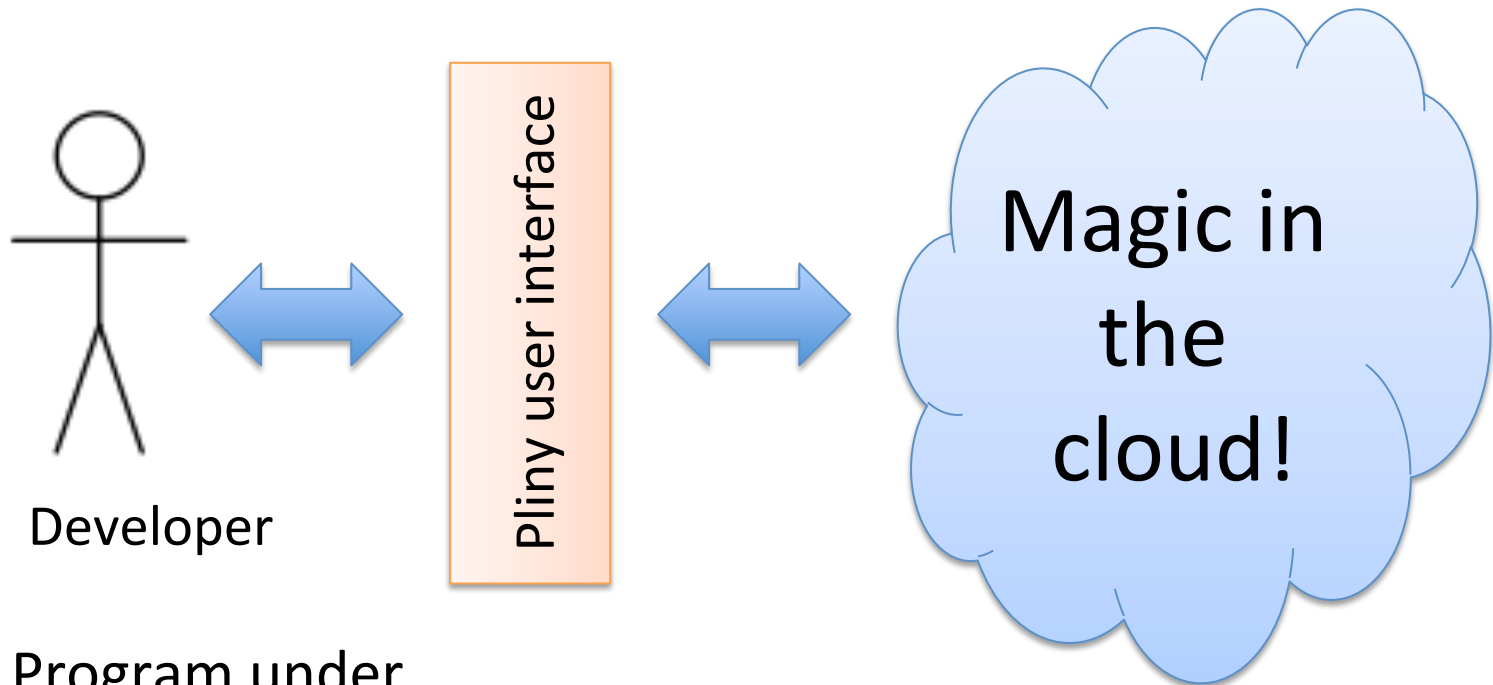
Pliny Reasoning Framework can leverage a wide range of languages, solvers, and user interfaces

PDB supports a wide range of user-programmable analytics ranging from data mining to machine learning

# Examples of Pliny's Open Architecture



# How can Similarity-Based Tools like Pliny Assist Porting of Applications to Exascale Platforms?



- *Inputs:* Program under development, performance tests
- *Outputs:* Identification of similar codes w/ transformations, selection of best transformations

How to extend the following to aid performance portability?

- Debugging
- Repair
- Program Synthesis

# Debugging, Repair, Synthesis for Performance

- *Debugging*: identify program points that do not match preferred transformations in code corpora (use profile information to focus on program regions of interest)
- *Repair*: identify local fixes to repair the performance “bugs”
- *Synthesis*: generate transformed versions of the program by implementing transformations suggested by similar codes in the corpus for similar platforms

Effectiveness of all of these techniques will depend on availability of code corpora with some/most well-tuned kernels/modules

# Other opportunities

- Add performance information from LCF runs of different codes
  - Give more weightage in similarity search to codes that are executed more often?
- Extend with use of natural language features from StackOverflow-like forums in similarity search
- Explore functionalities provided by past projects like Klonos
  - e.g., classification of subroutines in CESM climate code
- Leverage provenance information to tailor support for different application domains and different platforms
- ...

# Pliny Summary

---

- We are building a new “big code” system from scratch for extracting and storing code features, mining them for information, and leveraging the mined data for program synthesis, verification, debugging, and repair
- We have completed multiple demonstrations of the initial Pliny components working together
- Our implementation is based on an open architecture, and we look forward to exploiting this technology to address performance tuning/portability challenges in a new age of software development for HPC!