Visualizing a Moving Target: A Design Study on Task Parallel Programs in the Presence of Evolving Data and Concerns

Katy Williams et al., Arizona University

CSPC 547 Information Visualization 2019, **Michael Kim**

Visualizing a Moving Target: A Design Study on Task Parallel Programs in the Presence of Evolving Data and Concerns

Phylanx
An Asynchronous Distributed Array Computing Toolkit

Visualizing a Moving Target: A Design Study on Task Parallel Programs in the Presence of Evolving Data and Concerns

Design study on in-flight development project
Phylanx

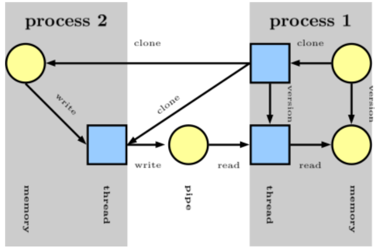
My Motivation

Why this paper?

An experience paper

Data is keep changing

Graph Visualization



What to look for?

Evolving Data and Concerns

How they deal with it

How they evaluate their system

Motivation : Chicken & Egg

Data vs. Analysis?

Analysis needed to decide what data to collect, but no data presented by domain experts

- **Lack of data availability and the domain users' needs**
- **Whether domain experts will use the visualization for will persist long enough to complete the study.**

Motivation: Task Parallel Programs

They want to accept the challenge not avoiding the pitfalls

Why they choose Task Parallel Programs?

Arizona University?

***Ease of access to a development environment**

High Complexity!

Correctness of Source Code
Parallel Libraries
The Input
Hardware Cluster
Policy
Asynchronous

...

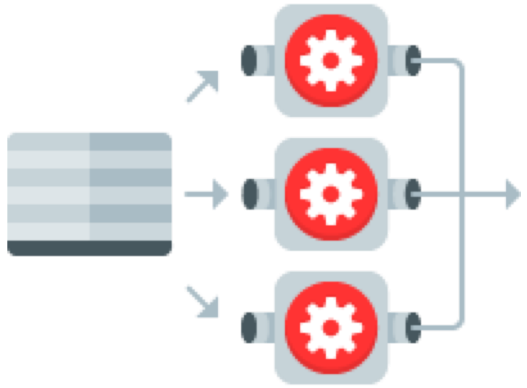
“Visualization Aid Needed for Debugging & Tuning”

Background: Phylanx

Python Code

TensorFlow

theano



PHYLANX

A DISTRIBUTED ARRAY TOOLKIT

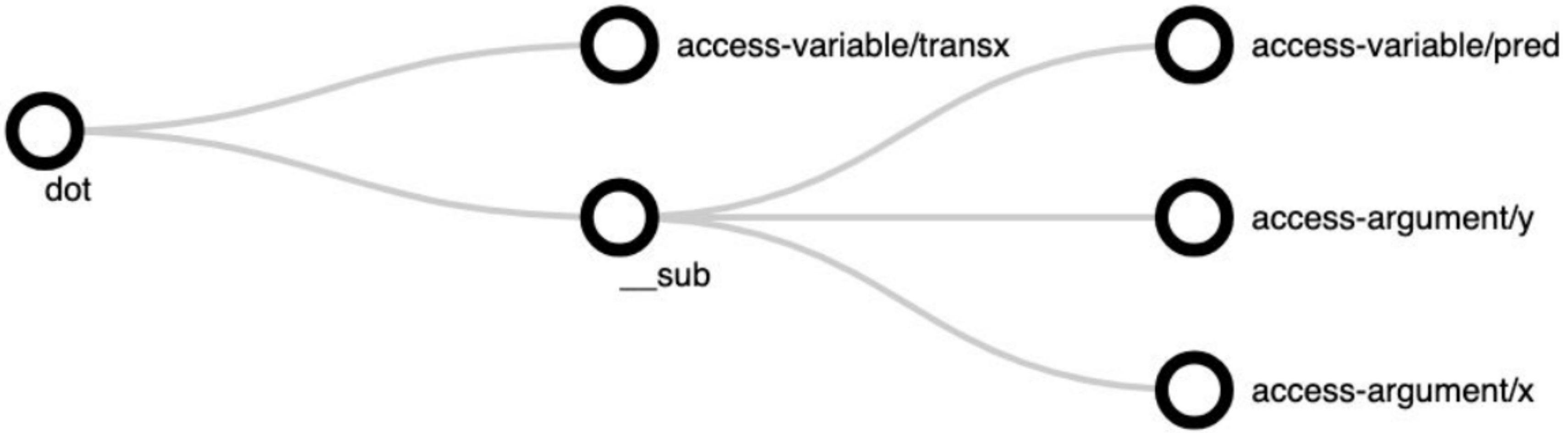
HPX
(C++ Async tasking)

Detail..

A platform … **for computations on distributed arrays for applied statistics on commodity cloud systems”**

The system will **optimize execution and data layout from of a user provided <u>expression graph</u>.**

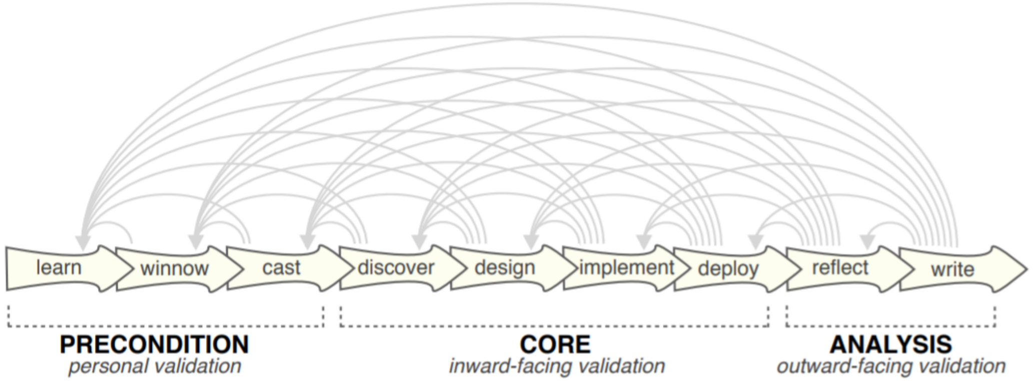
Background: Expression Graph



transx * (pred - y - x)

Node: Task / Primitive
Edge : Dependency

Design Methodology



Pitfalls

PF-4 : No Real Data Available
PF-10 : No Real/Important/Recurring Task.
PF-20 : Premature Design Commitment

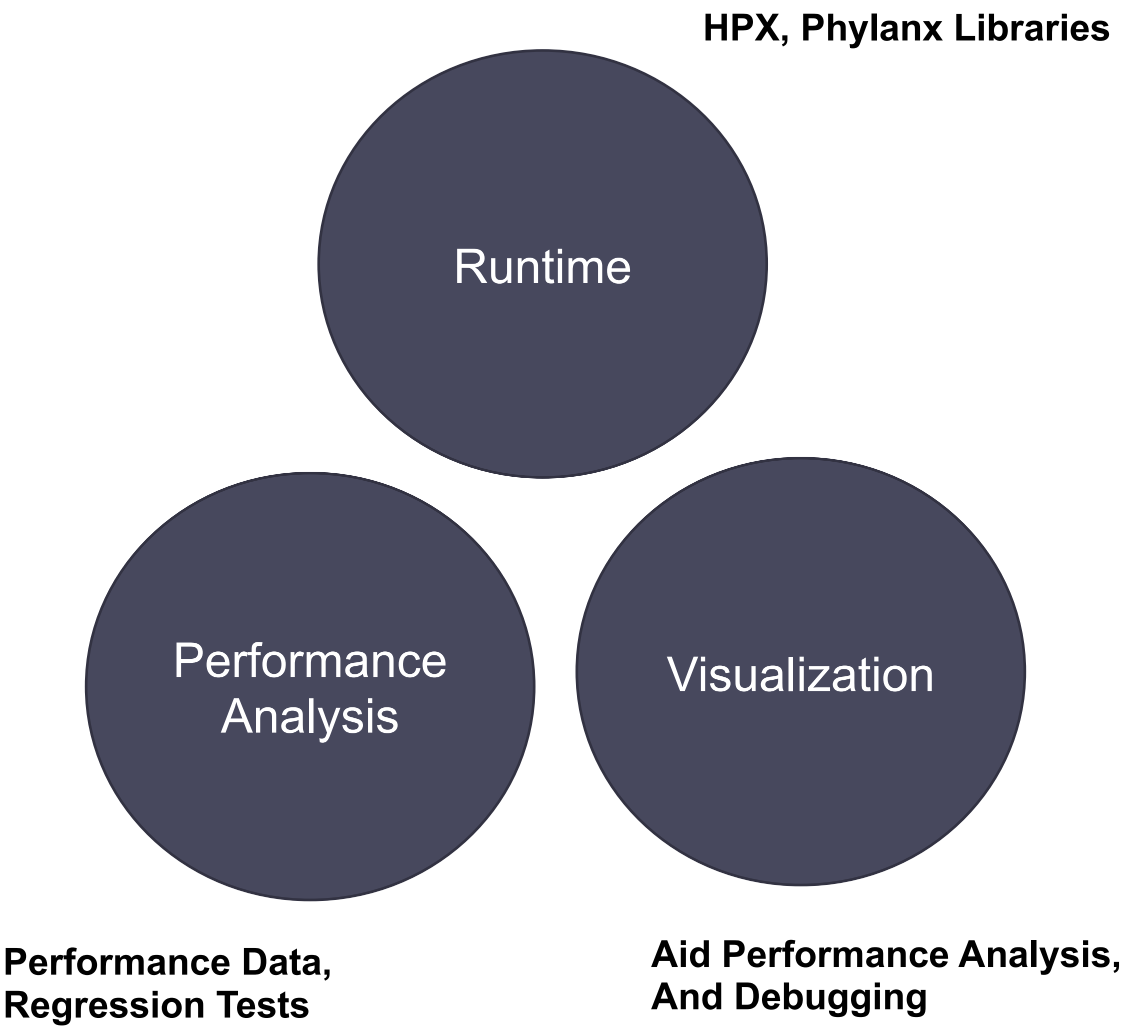
* “Design Study Methodology: Reflections from the Trenches and the Stacks”

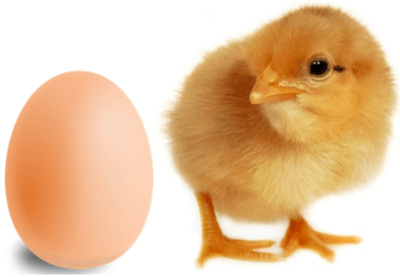
- “Communication, shared interest in the data collection problem
- The identification of key recurring abstract structures

Iterative process

Cast stage: Observed refinements, where deployments and conversations with domain experts exposed deeper insights into the various roles that they play in practice.

Methodology: Winnowing Pitfall


HPX, Phylanx Libraries
Runtime
Performance Analysis
Visualization
Performance Data, Regression Tests
Aid Performance Analysis, And Debugging

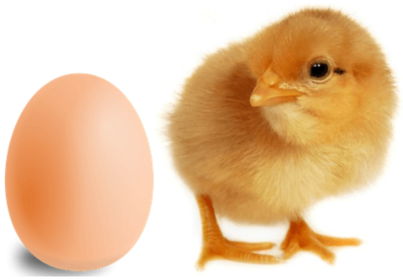


Winnowing Pitfall

PF-4: No Real Data Available (Yet). During the project, the structure of the data and the format of the data have been evolving. Other potential sources of data are not yet instrumented.

PF-10: No Real/Important/Recurring Task. The fact that the data is in flux means tasks involving that data are also in flux. Furthermore, as Phylanx is developing rapidly, the concerns of the team members change over time, affecting their higher-level goals.

Methodology: Example Pitfall Solution



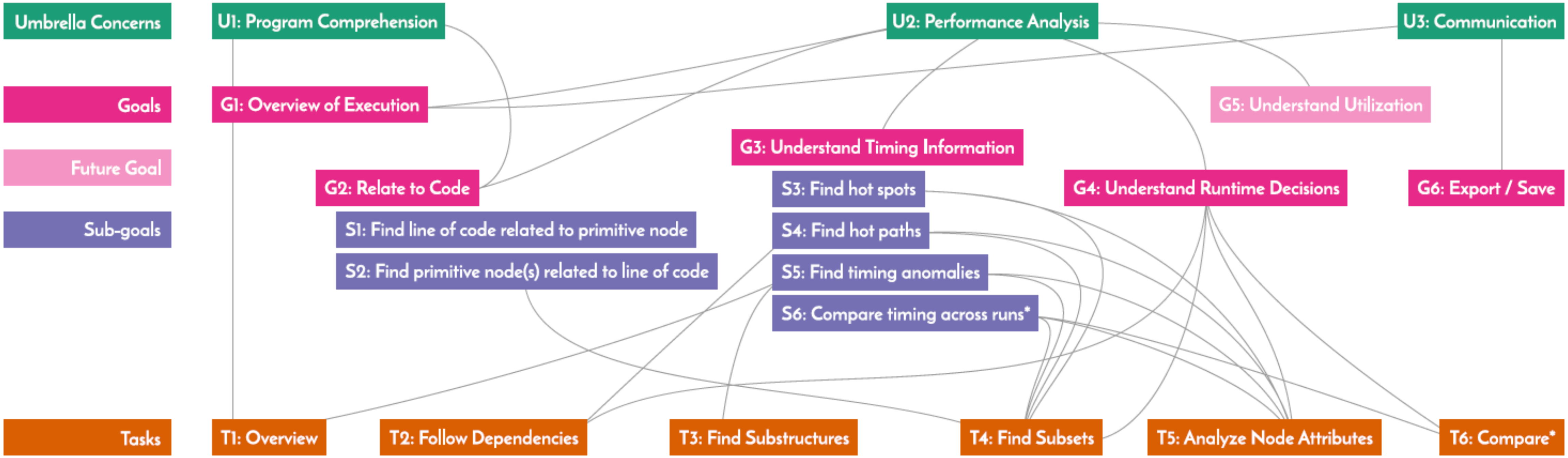
“Communication”

1) Identification and availability of meaningful preliminary data.
2) Strong interpersonal relationships.
3) Overarching goal of the project did not change.
4) Visualization considered a deliverable by entire project.

- Weekly report, “The incorporation of visualization as a project-wide outcome underscores the continuing approval and enthusiasm communicated by project gatekeepers”

- Through the present, we created 152 note files with a mean 2800 characters per file.

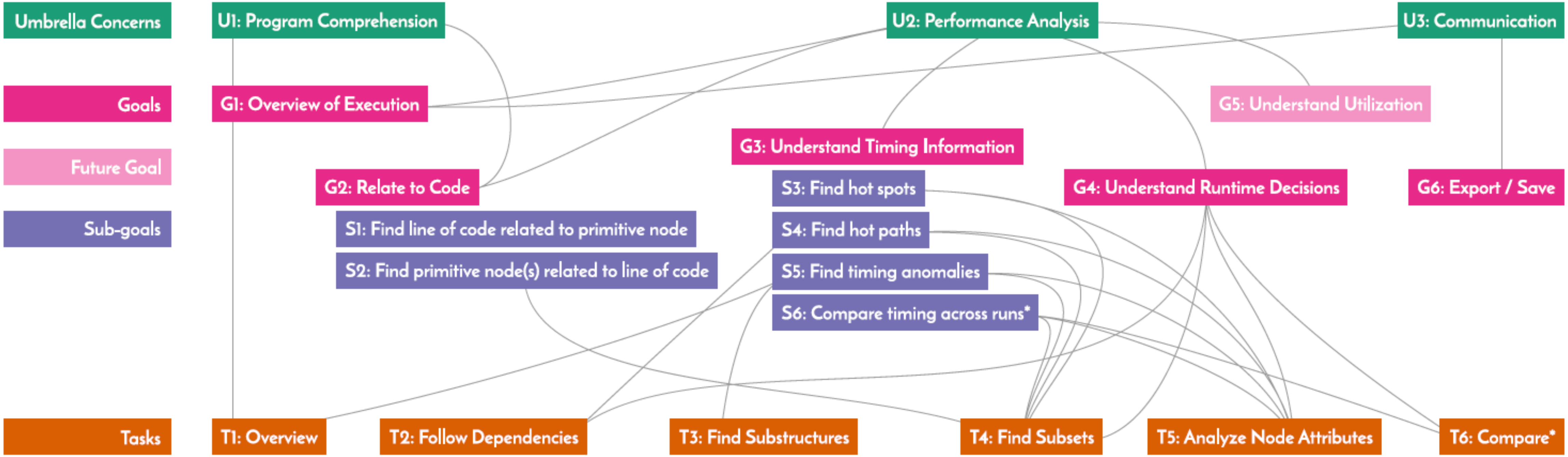
Task Abstraction: Goal-Task lattice-1


Umbrella Concerns
Goals
Future Goal
Sub-goals
Tasks
U1: Program Comprehension
U2: Performance Analysis
U3: Communication
G1: Overview of Execution
G5: Understand Utilization
G3: Understand Timing Information
G2: Relate to Code
S3: Find hot spots
G4: Understand Runtime Decisions
G6: Export / Save
S1: Find line of code related to primitive node
S4: Find hot paths
S2: Find primitive node(s) related to line of code
S5: Find timing anomalies
S6: Compare timing across runs*
T1: Overview
T2: Follow Dependencies
T3: Find Substructures
T4: Find Subsets
T5: Analyze Node Attributes
T6: Compare*

Tasks were derived from above: “Goal-Task lattice”

U1 Program Comprehension: What happens during program execution “Mental Model”
U2 Performance Analysis: Understanding and improving the performance of a given application
U3 Communication: Create figures to help explain their own research in publications.

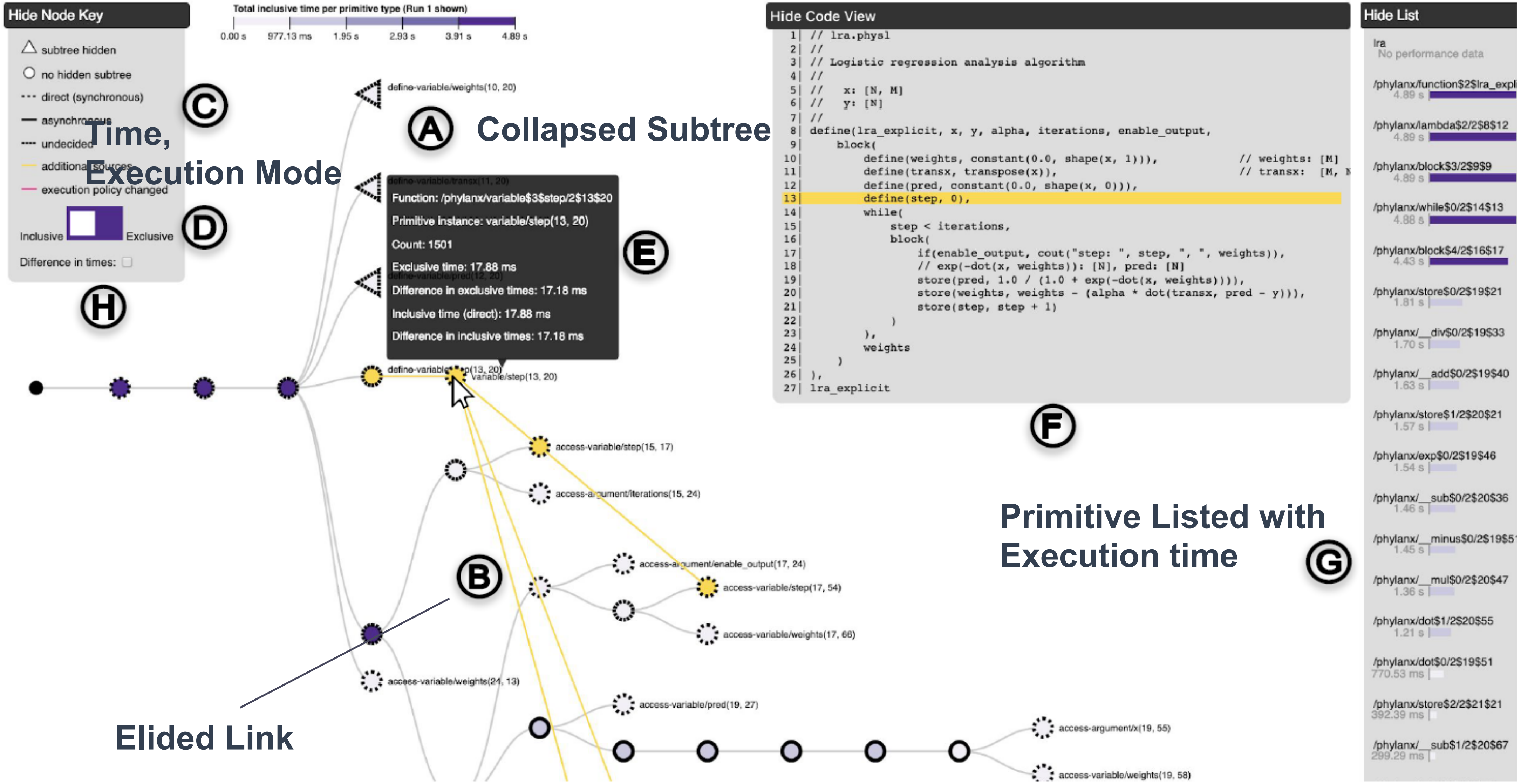
Task Abstraction: Goal-Task lattice-2


Umbrella Concerns
Goals
Future Goal
Sub-goals
Tasks
U1: Program Comprehension
U2: Performance Analysis
U3: Communication
G1: Overview of Execution
G5: Understand Utilization
G3: Understand Timing Information
G2: Relate to Code
S3: Find hot spots
G4: Understand Runtime Decisions
G6: Export / Save
S1: Find line of code related to primitive node
S4: Find hot paths
S2: Find primitive node(s) related to line of code
S5: Find timing anomalies
S6: Compare timing across runs*
T1: Overview
T2: Follow Dependencies
T3: Find Substructures
T4: Find Subsets
T5: Analyze Node Attributes
T6: Compare*

G1 Overview of Execution: Gaining a graph overview (T1) + Following dependencies (T2) + Finding substructures (T3)
G2 Relate to Code: Finding a subset of nodes (T4)
G3 Understand Timing Information: Finding a subset of nodes (T4) + Analyzing attribute data of those nodes (T5) + Following dependencies (T2 for Hot Paths) + Find Sub-structures (T3 for timing anomalies) + comparing attribute data (T6 for Comparison)

…

Visualization Design: Atria



Hide Node Key
△ subtree hidden
○ no hidden subtree
direct (synchronous)
asynchronous
undecided
additional sources
execution policy changed
Inclusive
Exclusive
Difference in times: □
Total inclusive time per primitive type (Run 1 shown)
0.00 s
977.13 ms
1.95 s
2.93 s
3.91 s
4.89 s
define-variable/weights(10, 20)
define-variable/transx(11, 20)
define-variable/pred(12, 20)
define-variable/step(13, 20)
variable/step(13, 20)
access-variable/step(15, 17)
access-argument/iterations(15, 24)
access-argument/enable_output(17, 24)
access-variable/step(17, 54)
access-variable/weights(17, 66)
access-variable/weights(21, 13)
access-variable/pred(19, 27)
access-argument/x(19, 55)
access-variable/weights(19, 58)
Hide Code View
Hide List
A
B
C
D
E
F
G
H

Time, Execution Mode

Collapsed Subtree

Elided Link

Primitive Listed with Execution time

Function: /phylanx/variable3step/2$13$20

Primitive instance: variable/step(13, 20)

Count: 1501

Exclusive time: 17.88 ms

Difference in exclusive times: 17.18 ms

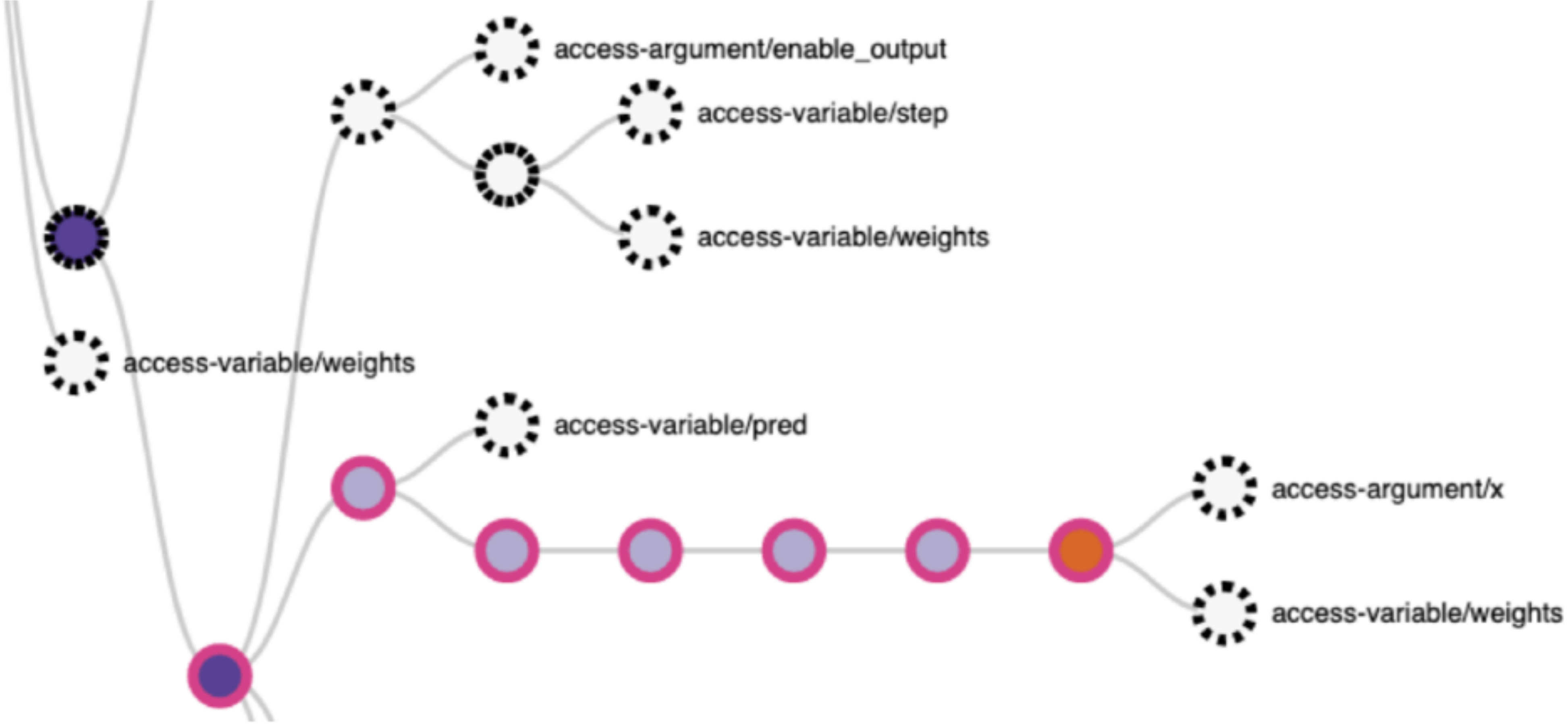
Inclusive time (direct): 17.88 ms

Difference in inclusive times: 17.18 ms

```
// lra.physl
//
// Logistic regression analysis algorithm
//
//    x: [N, M]
//    y: [N]
//
define(lra_explicit, x, y, alpha, iterations, enable_output,
    block(
        define(weights, constant(0.0, shape(x, 1))),            // weights: [M]
        define(transx, transpose(x)),                           // transx:  [M, N
        define(pred, constant(0.0, shape(x, 0))),
        define(step, 0),
        while(
            step < iterations,
            block(
                if(enable_output, cout("step: ", step, ", ", weights)),
                // exp(-dot(x, weights)): [N], pred: [N]
                store(pred, 1.0 / (1.0 + exp(-dot(x, weights)))),
                store(weights, weights - (alpha * dot(transx, pred - y))),
                store(step, step + 1)
            )
        ),
        weights
    )
),
lra_explicit
```

lra	No performance data
/phylanx/function2lra_expl	4.89 s
/phylanx/lambda$2/2$8$12	4.89 s
/phylanx/block$3/2$9$9	4.89 s
/phylanx/while$0/2$14$13	4.88 s
/phylanx/block$4/2$16$17	4.43 s
/phylanx/store$0/2$19$21	1.81 s
/phylanx/__div$0/2$19$33	1.70 s
/phylanx/__add$0/2$19$40	1.63 s
/phylanx/store$1/2$20$21	1.57 s
/phylanx/exp$0/2$19$46	1.54 s
/phylanx/__sub$0/2$20$36	1.46 s
/phylanx/__minus$0/2$19$5	1.45 s
/phylanx/__mul$0/2$20$47	1.36 s
/phylanx/dot$1/2$20$55	1.21 s
/phylanx/dot$0/2$19$51	770.53 ms
/phylanx/store$2/2$21$21	392.39 ms
/phylanx/__sub$1/2$20$67	299.29 ms

Visualization Design: Atria



Comparison between two runs of the same application with different policies.
Pink-outlined nodes indicate a difference in execution mode between two runs.
The orange node ran slower after the policy change.

Evaluation: Case Study, Task based Evaluation

-7 People (R4-R10) participated with distinct profile (~ R10 no prior experience)
-Case Study, Task abstraction based evaluation

[Atria Case Study] : Usage pattern, feedback

- "He reported using the visualization on average once a week, more frequently when actively debugging"

- "When explaining his workflow to us, R3 said "Also it's that I want to be able to visualize it [the algorithm], just seeing it implants it in my mind." He explained that he is a visual person and Atria makes it easier to think about the problem

[Task based evaluation] : Time takes, what difficulties and questions they had

L1: Find a primitive that takes a lot of time. (G3)
How long does it take without its children? With? (G3)
L2: Find a primitive that is executed synchronously (G4)
L3: Find a primitive that is executed asynchronously (G4)
L4: Find a primitive that is repeated in the code (G1)
C1: Which run was slower? (G3),
***Why might it have been slower? (G1, G2, G3, G4)**
C2: Find a primitive that changed execution mode. (G4) Explain the change. (G4)
=> "complete the L1 tasks within seconds"
=> R7 suggested that since the store took a lot of time, the program might be memory-intensive

Evaluation: Interview

[Interview] : Based on task evaluation

“Regarding utility, two participants said they didn’t know whether the features would be helpful or not (R6, R9).”

“Suggestions for improvement included differentiating primitive types (e.g., variables, functions, control-flow) (R6, R7)”

“Access to timing data (P4,P5, P7), the linked code view (P4, P5, P8), the comparison view (P4, P5, P9), and links between dependencies (P5, P7, P8)”

Lessons Learned

1) For “moving target”, seeking to satisfy rather optimize it

- PF-10, No Real/Important/Recurring Task

- PF-20, Premature Design Commitment

“Our rapid deployments often contained UI bugs”

2) Task analysis and long-term corpus of notes help clamp down on reactivity

3) Rapid changes combined with multiple deployment targets incur a maintenance burden

4) Both the visualization and the design study process aided our collaborators in accomplishing their goals and helped establish a culture of data review

THANK YOU