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

Katy Williams, Alex Bigelow, and Kate Isaacs

1 Introduction

We provide further detail on our collaborators involved with Atria’s design and the participants in our evaluation. We then provide more figures showing how Atria was used in practice and our evaluation sessions.

2 Collaborators and Participants

	Runtime Team	Performance Analysis Team	Gatekeeper	Front-line Analyst	Fellow Tool Builder	Evaluation participant	Prior familiarity	Influenced design	Deployment use
PM			●	●	●		●	●	D1, D7
P1		●	●	●	●		●	●	D2
P2		●		●	●		●	●	D1
R1	●		●	●	●		●	●	D1
R2	●		●	●	●		●	●	D5, D6
R3	●			●	●	●	●	●	D1
R4	●			●	●	●	●	●	D3
R5	●			●	●	●	●		
R6	●			●	●	●	●		
R7	●			●	●	●	●	●	
R8	●			●	●	●			
R9	●			●	●	●			
R10	●			●	●	●			
R11	●			●	●		●	●	D4, D6

Fig. 1. Team members involved in design or evaluation and their roles. Everyone had multiple roles, acted as a front-line analyst, and was a fellow tool builder.

The Phylanx project is divided into three teams: the Runtime Team, the Performance Analysis Team, and the Visualization Team. The Runtime Team is the largest with a principal investigator (PI), local project manager, several scientists, and student researchers. The Performance Analysis and Visualization teams each had a PI, student, and a scientist.

Each team resides at a different institution and time zone. In-person meetings occurred a few times a year, usually at the Runtime Team’s institution or the Supercomputing conference. There were weekly teleconferences with the team regarding all project goals (not only visualization). Visualization-focused video conference meetings could be scheduled on request, usually when the Visualization Team would demonstrate new features (new design iterations).

Figure 1 lists all Runtime and Performance Analysis team members who were involved in Atria’s design process, used Atria deployments, or who were involved in evaluation sessions. The Phylanx program manager is also listed.

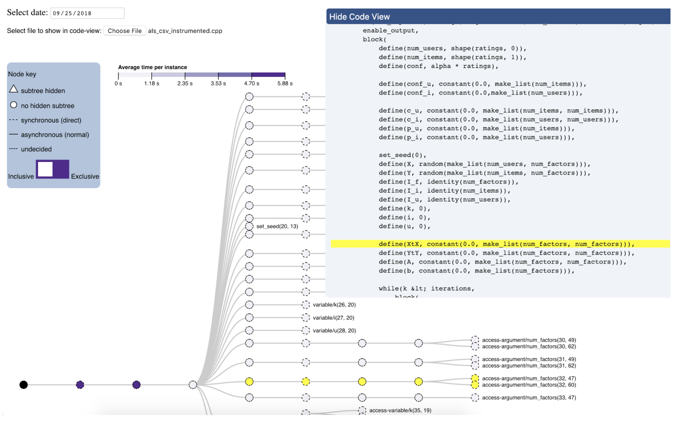
Most non-student team members have formal training in computer science or significant experience in scientific computing. Student team members were enrolled in either engineering or computer science graduate programs. Most team members, including the program and project managers, contributed code to the project, with the exception of newer students, such as R10, who were ramping up and new to the specific technologies of the project.

3 Deployments

Atria was deployed in several different ways, with design variations adapted to diverse, evolving needs.

Initially our data was output to the application’s standard out along with other debug output. Users had to extract Atria-specific data from the text dump manually. The first deployment (D1) was designed to serve data piped from the command line for generic *ad-hoc*, as-needed visualization. This deployment required users to clone and keep updated their own version of Atria. This version has been used extensively by R3, but also on occasion by P2 and R1. We also hosted our own version with fixed data as a demonstration site to communicate with the rest of the team. Additionally, PM originally used this version sporadically as a communication tool to describe how Phylanx worked to others outside the project.

A variant with a date selector was designed and deployed (D2) to augment a series of nightly regression tests that P1 had generated, to enable retroactive inspection, shown in Figure 2. Atria was then integrated into the nightly regression reports, allowing users to examine the previous night, any already collected regression data, or a comparison between two dates.


Select date: 09/25/2018
Select file to show in code-view:
Choose File
als_csv_instrumented.cpp
Hide Code View
Node key
subtree hidden
no hidden subtree
synchronous (direct)
asynchronous (normal)
undecided
Inclusive
Exclusive
Average time per instance
0 s
1.18 s
2.35 s
3.53 s
4.70 s
5.88 s
set_seed(20, 13)
variable/k(26, 20)
variable/i(27, 20)
variable/u(28, 20)
access-argument/num_factors(30, 49)
access-argument/num_factors(30, 62)
access-argument/num_factors(31, 49)
access-argument/num_factors(31, 62)
access-argument/num_factors(32, 47)
access-argument/num_factors(32, 60)
access-argument/num_factors(33, 47)
access-variable/k(35, 19)

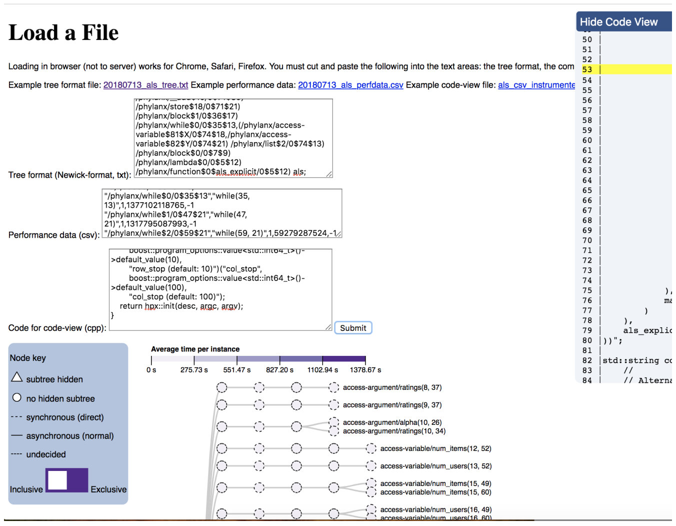
```
        enable_output,
        block(
            define(num_users, shape(ratings, 0)),
            define(num_items, shape(ratings, 1)),
            define(conf, alpha * ratings),

            define(conf_u, constant(0.0, make_list(num_items))),
            define(conf_i, constant(0.0,make_list(num_users))),

            define(c_u, constant(0.0, make_list(num_items, num_items))),
            define(c_i, constant(0.0, make_list(num_users, num_users))),
            define(p_u, constant(0.0, make_list(num_items))),
            define(p_i, constant(0.0, make_list(num_users))),

            set_seed(0),
            define(X, random(make_list(num_users, num_factors))),
            define(Y, random(make_list(num_items, num_factors))),
            define(I_f, identity(num_factors)),
            define(I_i, identity(num_items)),
            define(I_u, identity(num_users)),
            define(k, 0),
            define(i, 0),
            define(u, 0),

            define(XtX, constant(0.0, make_list(num_factors, num_factors))),
            define(YtY, constant(0.0, make_list(num_factors, num_factors))),
            define(A, constant(0.0, make_list(num_factors, num_factors))),
            define(b, constant(0.0, make_list(num_factors))),

            while(k &lt; iterations,
```

Fig. 2. A view of the deployment created for retroactive inspection of nightly regression tests (D2), including a calendar selector at the top-left.

• *Katy Williams, Alex Bigelow, and Kate Isaacs are with the University of Arizona. E-mails: kawilliams,alexrbigelow,kisaacs@email.arizona.edu.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

To streamline the setup burden of D1, we created another deployment (D3) which we hosted at visualization team's institution. This version allowed users to paste data from the standard out data dump into a text area, shown in Figure 3. R3 used this version when heavily examining one of the Phylanx applications, but as their work shifted from that application, is not using it now. Around the same time, R11 requested a self-contained, serverless HTML version of D1 with embedded data (D4).



Load a File

Loading in browser (not to server) works for Chrome, Safari, Firefox. You must cut and paste the following into the text areas: the tree format, the com

Example tree format file: 20180713_als_tree.txt Example performance data: 20180713_als_perfdata.csv Example code-view file: als_csv_instrumente

Tree format (Newick-format, txt):

```
/phylanx/store$18/0$71$21)
/phylanx/block$1/0$36$17)
/phylanx/while$0/0$35$13,(/phylanx/access-
variable$81$X/0$74$18,/phylanx/access-
variable$82$Y/0$74$21) /phylanx/list$2/0$74$13)
/phylanx/block$0/0$7$9)
/phylanx/lambda$0/0$5$12)
/phylanx/function$0$als_explicit/0$5$12) als;
```

Performance data (csv):

```
"/phylanx/while$0/0$35$13","while(35,
13)",1,1377102118765,-1
"/phylanx/while$1/0$47$21","while(47,
21)",1,1317795087993,-1
"/phylanx/while$2/0$59$21","while(59, 21)",1,59279287524,-1
```

Code for code-view (cpp):

```
>default_value(10),
      "row_stop (default: 10)")("col_stop",
      boost::program_options::value<std::int64_t>()-
>default_value(100),
      "col_stop (default: 100)");
   return hpx::init(desc, argc, argv);
}
```

Submit

Hide Code View

```
                    ),
                    ma
               )
     ),
     als_explic
))";

std::string co
     //
     // Alterna
```

Node key

△ subtree hidden

○ no hidden subtree

--- synchronous (direct)

— asynchronous (normal)

---- undecided

Inclusive Exclusive

Average time per instance
0 s
275.73 s
551.47 s
827.20 s
1102.94 s
1378.67 s
access-argument/ratings(8, 37)
access-argument/ratings(9, 37)
access-argument/alpha(10, 26)
access-argument/ratings(10, 34)
access-variable/num_items(12, 52)
access-variable/num_users(13, 52)
access-variable/num_items(15, 49)
access-variable/num_items(15, 60)
access-variable/num_users(16, 49)

Fig. 3. A view of the deployment hosted at the authors' institution, for cases where ports can not be opened publicly (D3), including text areas for pasting data.

At the request of the project manager, R2, we created a version with a specific dataset to work with Jupyter Notebook (D5) for demonstrations at the 2018 Supercomputing conference (SC18). This version contained only the main tree view, no code view or tooltip. In addition to R2, we also used this version, as well as D1, for demonstrations at SC18.

A persistent goal had been a more general Jupyter Notebook deployment. Having overcome technological considerations to create the SC18 demo, we iterated on the design, moving the code view and tool tips, resulting in (D6), shown in Figure 5 in the paper. We incorporated this version into a new demonstration notebook, both for the team members who give general demonstrations (R2, PM) and as an example for potential Phylanx users.

Finally, PM created a variant for use and demonstration at a secure facility (D7). This version is undergoing review and could not be shared with the authors.

4 Atria Figures from Evaluations

At the start of each evaluation session, we introduced the participant to Atria by walking them through a tree of a factorial code snippet, shown in Figure 4. The familiarity of the code and the small size of the tree made it a tractable example of Atria's features.

After the factorial demonstration, we asked the user to perform a series of tasks, as listed in Section 7.2.1. The single tree view is shown in Figure 5.

For task L1 ("Find a primitive that takes a lot of time") the expected workflow was to visually scan for the nodes with the most saturated color. To answer the follow-up questions ("How long does it take without its children? With?") the user needed to hover the mouse over the node to read the tooltip, which displayed the exclusive time ("without its children") and the inclusive time ("with").

In task L2 ("Find a primitive that is executed synchronously") the user visually scanned the tree for a node with a dashed border. There are many instances of nodes with dashed borders in the tree so participants quickly found a correct node. Some participants had difficulty distinguishing between the close-dashed and spaced-dashed lines.

Task L3 ("Find a primitive that is executed asynchronously") was similar to task L2. However, there was only one such node in this example, so some participants had difficult finding this single node with a solid border.

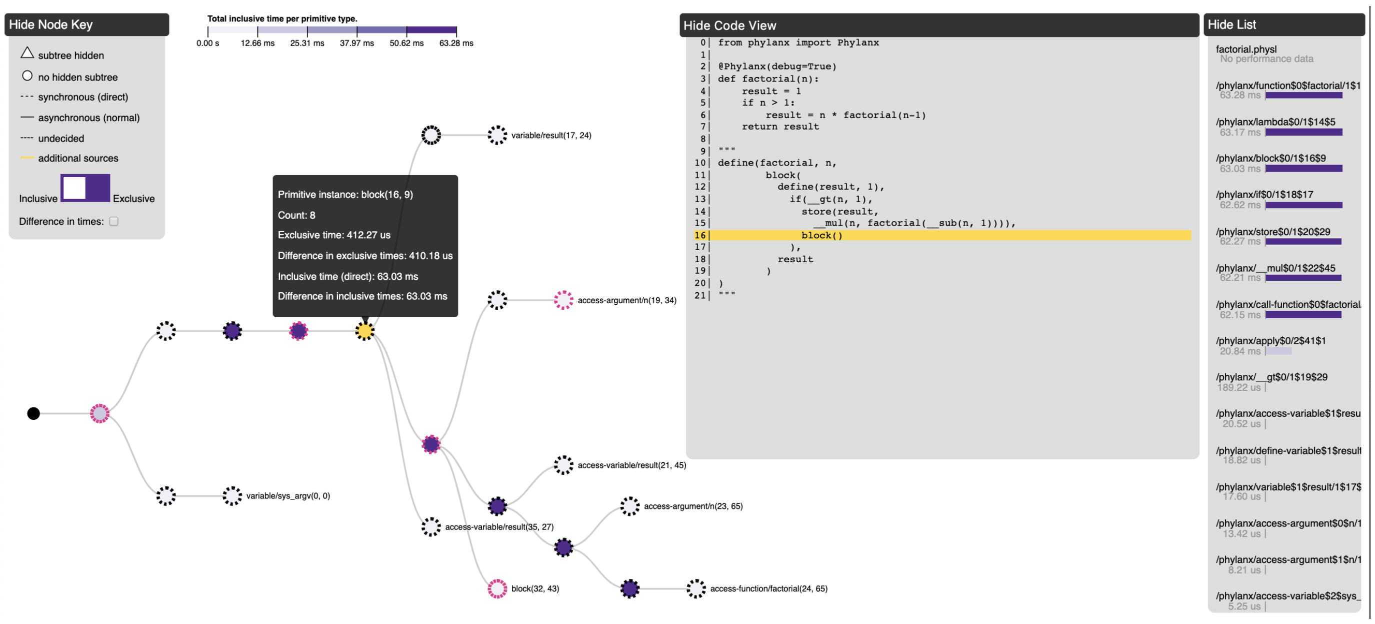
Task L4 ("Find a primitive that is repeated in the code") required more interaction with Atria besides simple visual scanning. Since the yellow lines that connect repeated primitives are only visible when the user hovers over the primitive, the users used their mouse to manually scan over nodes in the tree and to discover if the primitive was reused.

Once the user completed tasks on a single tree, we presented them with the comparison tree view, as show in Figure 6. We asked the user to perform a series of tasks that focused on comparing two datasets, as listed in Section 7.2.1.

For task C1 ("Which run was slower?") the user visually scanned the entire tree to collect information about the hue and saturation of the nodes as a whole. A tree with more saturated purple nodes indicates that the net change in time increased from the first run to the second run, therefore the differences in the second code led to a faster run. Likewise, a tree with more orange nodes indicates that the first run of the code was faster than the second run.

We only presented task C1* to users with a performance analysis background. This question ("Why might it have been slower?") required the user to investigate highly saturated nodes of the slower run's color and reason about the primitive represented by the node.

Users accomplished task C2 ("Find a primitive that changed execution mode") by visually scanning the tree for nodes with a pink border. Most users confirmed their answer by reading the tooltip.

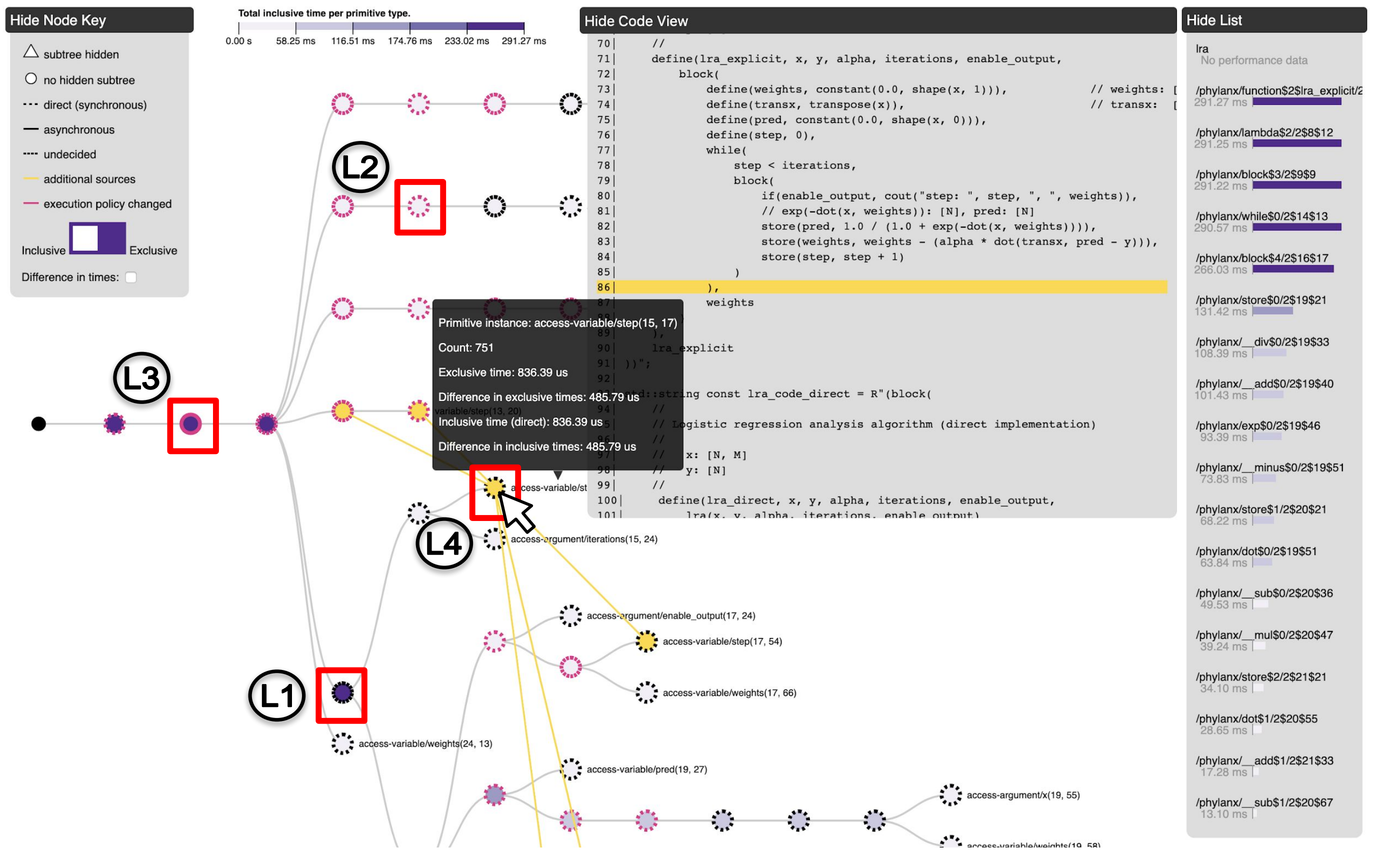


Primitive instance: block(16, 9)
Count: 8
Exclusive time: 412.27 us
Difference in exclusive times: 410.18 us
Inclusive time (direct): 63.03 ms
Difference in inclusive times: 63.03 ms

```
from phylanx import Phylanx

@Phylanx(debug=True)
def factorial(n):
    result = 1
    if n > 1:
        result = n * factorial(n-1)
    return result

"""
define(factorial, n,
        block(
          define(result, 1),
            if(__gt(n, 1),
              store(result,
                __mul(n, factorial(__sub(n, 1)))),
              block()
            ),
          result
        )
)
"""
```

factorial.physl
No performance data
/phylanx/function$0$factorial/1$1
63.28 ms
/phylanx/lambda$0/1$14$5
63.17 ms
/phylanx/block$0/1$16$9
63.03 ms
/phylanx/if$0/1$18$17
62.62 ms
/phylanx/store$0/1$20$29
62.27 ms
/phylanx/__mul$0/1$22$45
62.21 ms
/phylanx/call-function0factorial
62.15 ms
/phylanx/apply$0/2$41$1
20.84 ms
/phylanx/__gt$0/1$19$29
189.22 us
/phylanx/access-variable1resu
20.52 us
/phylanx/define-variable1result
18.82 us
/phylanx/variable1result/1$17$
17.60 us
/phylanx/access-argument0n/1
13.42 us
/phylanx/access-argument1n/1
8.21 us
/phylanx/access-variable2sys_
5.25 us

Fig. 4. A view of the demo tree we showed participant at the start of each Evaluation Session.

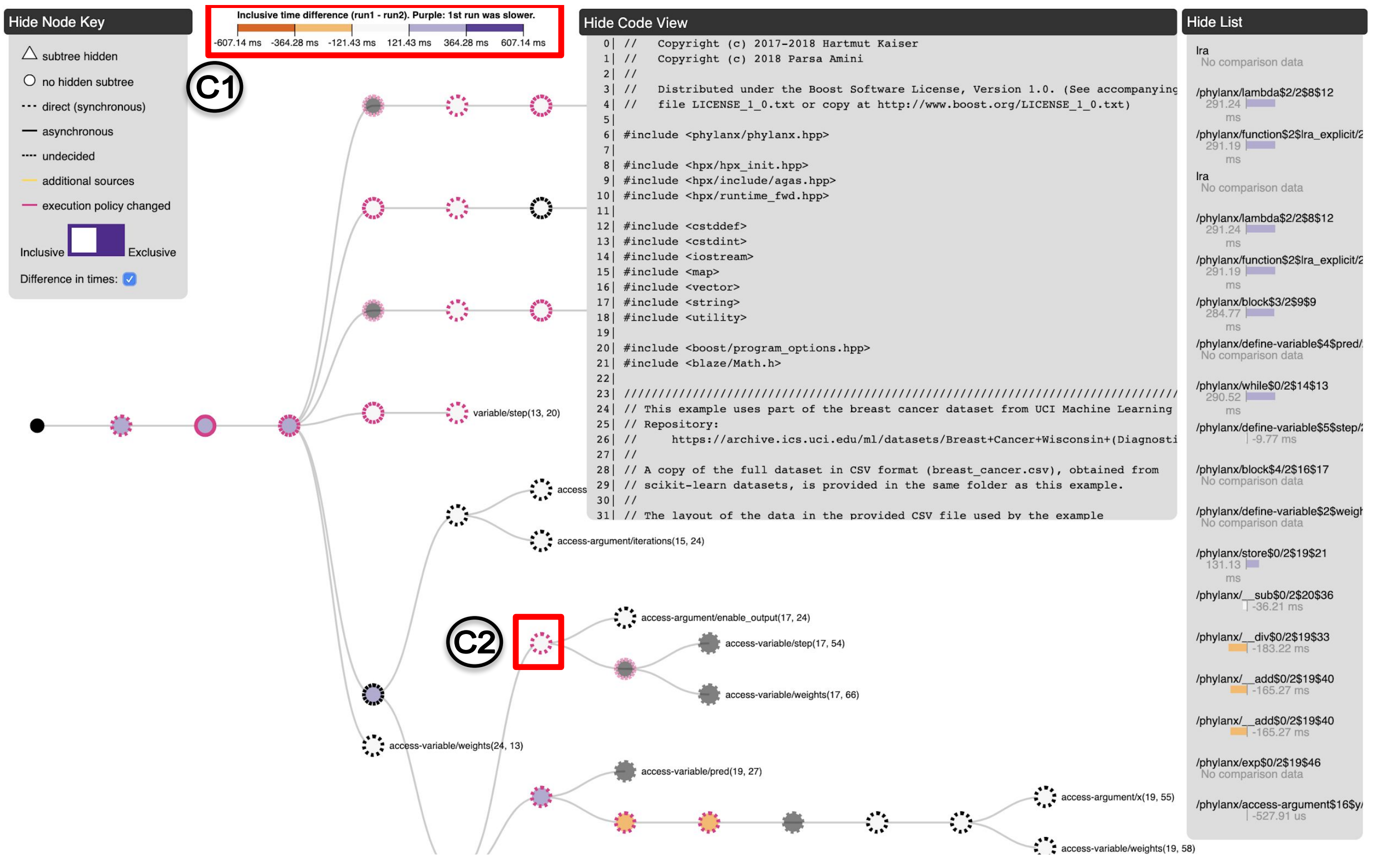

Hide Node Key
subtree hidden
no hidden subtree
direct (synchronous)
asynchronous
undecided
additional sources
execution policy changed
Inclusive
Exclusive
Difference in times:
Total inclusive time per primitive type.
0.00 s
58.25 ms
116.51 ms
174.76 ms
233.02 ms
291.27 ms
L1
L2
L3
L4
access-variable/step(13, 20)
access-argument/iterations(15, 24)
access-argument/enable_output(17, 24)
access-variable/step(17, 54)
access-variable/weights(17, 66)
access-variable/weights(24, 13)
access-variable/pred(19, 27)
access-argument/x(19, 55)
access-variable/weights(19, 58)
Hide Code View
Hide List

Primitive instance: access-variable/step(15, 17)
Count: 751
Exclusive time: 836.39 us
Difference in exclusive times: 485.79 us
Inclusive time (direct): 836.39 us
Difference in inclusive times: 485.79 us

```
    //
    define(lra_explicit, x, y, alpha, iterations, enable_output,
        block(
            define(weights, constant(0.0, shape(x, 1))),           // weights: [
            define(transx, transpose(x)),                          // transx:  [
            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
))";

std::string const lra_code_direct = R"(block(
    //
    // Logistic regression analysis algorithm (direct implementation)
    //
    //   x: [N, M]
    //   y: [N]
    //
     define(lra_direct, x, y, alpha, iterations, enable_output,
         lra(x, y, alpha, iterations, enable_output)
```

lra
No performance data
/phylanx/function2lra_explicit/2
291.27 ms
/phylanx/lambda$2/2$8$12
291.25 ms
/phylanx/block$3/2$9$9
291.22 ms
/phylanx/while$0/2$14$13
290.57 ms
/phylanx/block$4/2$16$17
266.03 ms
/phylanx/store$0/2$19$21
131.42 ms
/phylanx/__div$0/2$19$33
108.39 ms
/phylanx/__add$0/2$19$40
101.43 ms
/phylanx/exp$0/2$19$46
93.39 ms
/phylanx/__minus$0/2$19$51
73.83 ms
/phylanx/store$1/2$20$21
68.22 ms
/phylanx/dot$0/2$19$51
63.84 ms
/phylanx/__sub$0/2$20$36
49.53 ms
/phylanx/__mul$0/2$20$47
39.24 ms
/phylanx/store$2/2$21$21
34.10 ms
/phylanx/dot$1/2$20$55
28.65 ms
/phylanx/__add$1/2$21$33
17.28 ms
/phylanx/__sub$1/2$20$67
13.10 ms

Fig. 5. A view of the single tree that participants used to perform our tasks during the evaluation sessions. The red boxes show a possible answer to each question presented in the task. Each box is labeled with its corresponding task, as they are enumerated in Section 7.2.1 (Tasks L1 - L4).


Hide Node Key
subtree hidden
no hidden subtree
direct (synchronous)
asynchronous
undecided
additional sources
execution policy changed
Inclusive
Exclusive
Difference in times: ✓
Inclusive time difference (run1 - run2). Purple: 1st run was slower.
-607.14 ms
-364.28 ms
-121.43 ms
121.43 ms
364.28 ms
607.14 ms
C1
C2
variable/step(13, 20)
access
access-argument/iterations(15, 24)
access-argument/enable_output(17, 24)
access-variable/step(17, 54)
access-variable/weights(17, 66)
access-variable/weights(24, 13)
access-variable/pred(19, 27)
access-argument/x(19, 55)
access-variable/weights(19, 58)
Hide Code View
Hide List

```
//   Copyright (c) 2017-2018 Hartmut Kaiser
//   Copyright (c) 2018 Parsa Amini
//
//   Distributed under the Boost Software License, Version 1.0. (See accompanying
//   file LICENSE_1_0.txt or copy at http://www.boost.org/LICENSE_1_0.txt)

#include <phylanx/phylanx.hpp>

#include <hpx/hpx_init.hpp>
#include <hpx/include/agas.hpp>
#include <hpx/runtime_fwd.hpp>

#include <cstddef>
#include <cstdint>
#include <iostream>
#include <map>
#include <vector>
#include <string>
#include <utility>

#include <boost/program_options.hpp>
#include <blaze/Math.h>

////////////////////////////////////////////////////////////////////////////////
// This example uses part of the breast cancer dataset from UCI Machine Learning
// Repository:
//     https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnosti
//
// A copy of the full dataset in CSV format (breast_cancer.csv), obtained from
// scikit-learn datasets, is provided in the same folder as this example.
//
// The layout of the data in the provided CSV file used by the example
```

lra	No comparison data
/phylanx/lambda$2/2$8$12	291.24 ms
/phylanx/function2lra_explicit/2	291.19 ms
lra	No comparison data
/phylanx/lambda$2/2$8$12	291.24 ms
/phylanx/function2lra_explicit/2	291.19 ms
/phylanx/block$3/2$9$9	284.77 ms
/phylanx/define-variable4pred/	No comparison data
/phylanx/while$0/2$14$13	290.52 ms
/phylanx/define-variable5step/2	-9.77 ms
/phylanx/block$4/2$16$17	No comparison data
/phylanx/define-variable2weigh	No comparison data
/phylanx/store$0/2$19$21	131.13 ms
/phylanx/__sub$0/2$20$36	-36.21 ms
/phylanx/__div$0/2$19$33	-183.22 ms
/phylanx/__add$0/2$19$40	-165.27 ms
/phylanx/__add$0/2$19$40	-165.27 ms
/phylanx/exp$0/2$19$46	No comparison data
/phylanx/access-argument16y/	-527.91 us

Fig. 6. A view of the comparison tree that participants used to perform our tasks during the evaluation sessions. The red boxes show a possible answer to each question presented in the task. Each box is labeled with its corresponding task, as they are enumerated in Section 7.2.1 (Tasks C1 and C2).