University of Basel

Managing Resources of Network Nodes Using Append-Only-Logs

Simon Laube, simon.laube@stud.unibas.ch

Advisor: Prof. Dr. Christian Tschudin, Supervisor: Fabrizio Parrillo

July 12, 2022

Content

Motivation

- Solar Community Network

Initial Goals

- Affordable Hardware
- Long Transmission Range (Wireless)
- Resilient Communication Protocol
  - Low Power Consumption
  - Low Storage Usage
- Hardware + Software $\rightarrow$ Proof-of-Concept

Thesis Focus

- Affordable Hardware
- Long Transmission Range (Wireless)
- **Resilient Communication Protocol**
  - **Low Power Consumption**
  - **Low Storage Usage**
- **Hardware + Software $\rightarrow$ Proof-of-Concept**

Resilient Communication Protocol

- TinySSB
  - Tiny Version of Secure Scuttlebutt (Peer-to-Peer Communication Protocol)
  - Append-Only-Logs
  - Trust Anchors and Chain of Trust

Feeds

- Everything Stored in Feeds
- Child Feeds
- Continuation Feeds

Limitations

- Revert Packets
- Delete Old Packets

Addition: Fork-Tree

- ...

Addition: Session-Tree

- ...

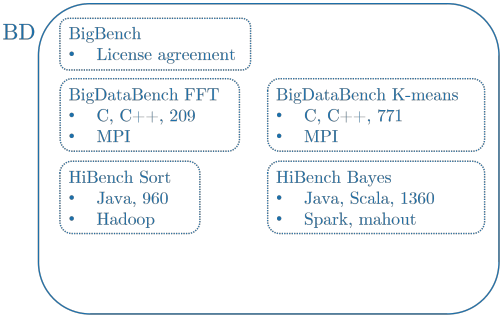
Development Setup

- Micropython
- Test Code on Computer (use UDP)
- Test Code on LoPy 4 (use LoRa)

Issues

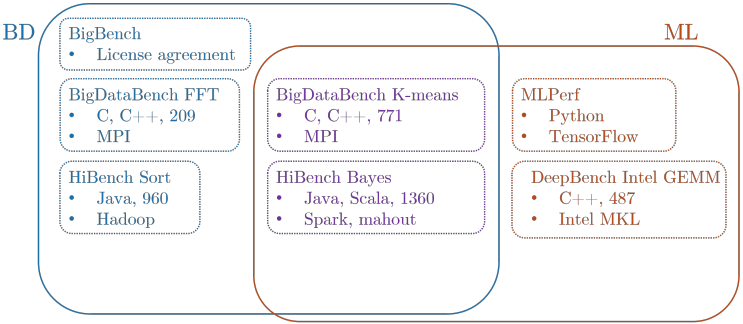
- Stack Overflows
  - Too much memory usage (not noticed on computer)
  - Format of LoPy 4 disk
- Testing on LoPy much slower (e.g. Ed25519)
- PyMakr Extension (Visual Studio, Atom)

Issues



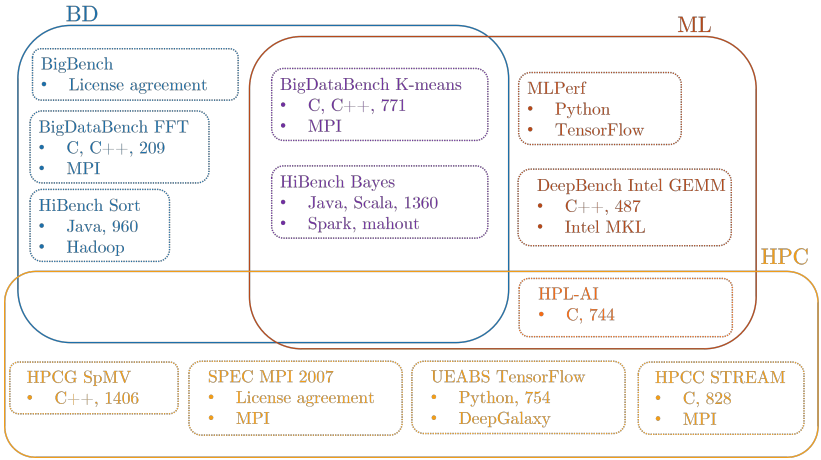
- Domain
- Benchmark
- Language
- LOC
- third-party software

Issues



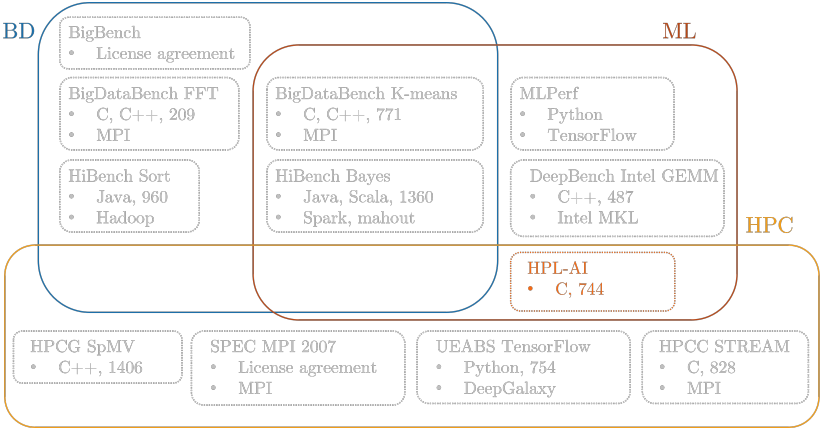
- Domain
- Benchmark
- Language
- LOC
- third-party software

Issues



- Domain
- Benchmark
- Language
- LOC
- third-party software

Issues



- Domain
- Benchmark
- Language
- LOC
- third-party software

HPL-AI

- Solving a linear system $Ax = b$ with LU decomposition.
- Generating random A and b double precision.
- Convert A and b to single precision.
- Decompose $A = LU$. Solve $Ly = b$ forwards and $Ux = y$ backwards.
- Convert x back to double precision and reinstate accuracy with generalized minimal residual method (GMRES).

Code snippets

Implemented function to convert matrix from double to single precision:

```
def convert_f64_to_f32(mat) {

    print(mat);
    i = 0;
    j = 0;
    print(mat[i,j]);

    # symmetric matrix, nrow returns row of type index
    # this is not working with for and 1:n, thus make si64
    n = as.si64(nrow(mat));

    # result matrix, type float
    m = fill(as.f32(0),n,n);
    print(m);

    # val = as.f32(sum(mat[seq(0,0,1),seq(0,0,1)]));
    # print(val);

    for (i in 0:(n - 1)) {
        for (j in 0:(n - 1)) {
            posi = seq(i,i,0);
            posj = seq(j,j,0);
            # sum used to extract single value of slice
            val = as.f32(sum(mat[posi,posj]));
            print(mat[posi,posj]);
            m[i,j] = fill(as.f32(1),1,1);
        }
    }

    return m;
}
```

Implemented function for LU decomposition:

```
# algorithm for LU decompositio-n: A = LU
def LU(A) {

    # symmetric matrix, nrow returns row of type index
    # this is not working with for and 1:n, thus make si64
    n = as.si64(nrow(A));
    LU = A;

    for (k in 0:(n - 2)) {
        for (i in (k):(n - 1)) {
            posk = seq(k,k,1);
            posi = seq(i,i,1);
            LU[posi,posk] = LU[posi,posk] / LU[posk,posk];
            for (j in k:(n+-1)) {
                posj = seq(j,j,1);
                LU[posi,posj] = LU[posi,posj] - LU[posk,posj] * LU[posi,posk];
            }
        }
    }

    return LU;
}
```

Reported DaphneDSL issues

GitHub	Issue	Description	Type
#350	Rbind row Check	rbind() checking number of rows instead of columns	Bug
?	Parsing $n - 1$	Parsing only works if there is a space after the $-$, i.e. $n- 1$	Bug
#351	Printing with variables	Printing with calculated variables behaving differently than with assigned variables	Bug
#352	Integer matrix multiplication	Integer vector multiplication not working	Bug
#353	Right indexing in loops	Right indexing with variables in loops is different from outside and is not working in functions	Bug
#354	Left indexing in loops	Left indexing with variables in loops is different from outside and is not working in functions	Bug
#355	Return from nrow()	retrun value from nrow() not usable in operation	Bug
#356	Type casting matrix	Casting on matrices not working	Bug
#357	typeof() function	function returning the type of an object	Nice to have
#358	Matrix literals	Simple way to create a not random value matrix	Nice to have

Code for issues: Bugs

Rbind row check:

```
c1 = fill(17,1,1);
c2 = fill(24,1,1);
c3 = fill(1,1,1);
c4 = fill(8,1,1);
c5 = fill(15,1,1);
r = cbind(c1,c2);
r = cbind(r,c3);
r = cbind(r,c4);
r = cbind(r,c5);

c1 = fill(23,1,1);
c2 = fill(5,1,1);
c3 = fill(7,1,1);
c4 = fill(14,1,1);
c5 = fill(16,1,1);
r2 = cbind(c1,c2);
r2 = cbind(r2,c3);
r2 = cbind(r2,c4);
r2 = cbind(r2,c5);

c1 = fill(4,1,1);
c2 = fill(6,1,1);
c3 = fill(13,1,1);
c4 = fill(20,1,1);
c5 = fill(22,1,1);
r3 = cbind(c1,c2);
r3 = cbind(r3,c3);
r3 = cbind(r3,c4);
r3 = cbind(r3,c5);

print(r);
print(r2);
print(r3);

r = rbind(r,r2);
print(r);
# here it checks number of rows instead of number of colums!!
r = rbind(r,r3);
# Output:
# Pass error: shape inference:
# inferNumColsFromArgs() requires that arguments have the same number
# of columns, but there is one with 2 and one with 1 columns
```

Parsing $n - 1$:

```
n=5;
print(n+1);
print(n - 1);
print(n- 1);
print(-1);
# not working
print(n-1);
```

Code for issues: Bugs

Printing with variables:

```
# working
time_convert = 200;
print("Time spend: " + time_convert + " seconds");
# this is not working correctly:
time_convert = 200 - 20;
print("Time spend:" + time_convert + " seconds");
# Output
# Time spend: 200 seconds
# 180Time spend: seconds
```

Integer matrix multiplication:

```
# working with doubles
d1 = reshape(seq(1.0,5.0,1.0),1,5);
print(d1);
d2 = reshape(seq(1.0,5.0,1.0),5,1);
print(d2);
print(d1 @ d2);

# not working with integer
i1 = reshape(seq(1,5,1),1,5);
print(i1);
i2 = reshape(seq(1,5,1),5,1);
print(i2);
print(i1 @ i2);
```

Code for issues: Bugs

Right indexing in loops:

```
m = rand(5,5,0.0,1.0,1,-1);
# working
i = 1;
print(m[i,i]);

# not working
for (i in 0:2) {
    # print(m[i,i]);
}

# working, but not the most obious way
for (i in 0:2) {
    pos_i = seq(i,i,1);
    print(m[pos_i,pos_i]);
}

# not working in functions at all
def print_mat(mat :matrix) {
    print(mat[0,0]);
}
print_mat(m);
```

Left indexing in loops:

```
# all working
A = rand(5,5,0.0,1.0,1,-1);
print(A);
A[0,0] = fill(10.0,1,1);
print(A);
B = rand(1,5,1.0,2.0,1,-1);
print(B);
A[4,:] = B;
print(A);
i = 3;
C = rand(1,4,2.0,3.0,1,-1);
A[i,1:] = C;
print(A);

for (i in 0:2) {
    # working
    A = rand(5,5,0.0,1.0,1,-1);
    A[0,0] = fill(10.0,1,1);
    # working
    B = rand(1,5,1.0,2.0,1,-1);
    A[4,:] = B;
    # not working
    C = rand(1,4,2.0,3.0,1,-1);
    A[i,1:] = C;
    # Output: JIT session error: Symbols not found:
}

# not working
def change_element(A) {
    i = 3;
    C = rand(1,4,2.0,3.0,1,-1);
    A[i,1:] = C;
    # Output: Parser error:
    # right indexing is only allowed on matrices and frames
    return A;
}
change_element(A);
```

Code for issues: Bugs

Return from nrow():

```
m = rand(5,5,0.0,1.0,1,-1);
n = nrow(m);
print(n); # working
print(n+1); # not working
# Output:
# error: 'daphne.ewAdd' op operand #0 must be matrix of numeric
# or placeholder for an unknown type values or numeric, but got 'index'
```

Type casting matrix:

```
# Issue 8: change value type of matrix
# ------------------------------------
# Even though mentioned in documentation, its not working

A = rand(5,5,0.0,1.0,1,-1);
print(A);
B = as.matrix.f32(A);
# Output:
# JIT session error: Symbols not found:
#[ _cast__DenseMatrix_float__DenseMatrix_double ]
```

Code for issues: Nice to have

typeof() function:

```
# type function for objects returning the type/typeof similar as in R

f = 5.0;
typeof(f); # --> f64
d = reshape(seq(1.0,5.0,1.0),5,1);
typeof(d1); # --> matrix double
i = reshape(seq(1,5,1),1,5);
typeof(i); # --> matrix int64
```

Matrix literals:

```
# create a simple matrix (like matlab) with some desired values
# or at least vector, as reshape exists (like in R)
# See Issue 1. from bug, for loc to generate 5x5 magic square

# matlab
m = [17 24 1 8 15, 23 57 14 16];
# R
v = c(17 2 4 1 8 15 23 5 7 14 16);
m = reshape(m,5,2)
# or with matrix command
m = matrix(17 2 4 1 8 15 23 5 7 14 16);
```

Another way: Using built in functionality

- Using DaphneDSL `as.` for casting and `solve` to solve linear system.

Another way: Using built in functionality

- Using DaphneDSL `as.` for casting and `solve` to solve linear system.
- Advantage: Few lines of code if available.

Another way: Using built in functionality

- Using DaphneDSL `as.` for casting and `solve` to solve linear system.
- Advantage: Few lines of code if available.
- Disadvantage: `solve` is not vectorized.

Another way: Using built in functionality

- Using DaphneDSL `as.` for casting and `solve` to solve linear system.
- Advantage: Few lines of code if available.
- Disadvantage: `solve` is not vectorized.
- Disadvantage: Not knowing what is actually happening inside DAPHNE

Next steps

- Find benchmarks implementable given the current state of DaphneDSL.
- Open new issues on GitHub for additional problems.

Challenges

- The timeline of the thesis is not matching the time horizon for solving issues

Challenges

- The timeline of the thesis is not matching the time horizon for solving issues
- DAPHNE is an ongoing project and things are not finished yet (e.g. documentation)

Challenges

- The timeline of the thesis is not matching the time horizon for solving issues
- DAPHNE is an ongoing project and things are not finished yet (e.g. documentation)
- Any questions or suggestions?