Pando: An Easy-to-Deploy P2P Volunteer Computing Platform for the Web

Erick Lavoie, Laurie Hendren, Frédéric Desprez

Leverage all your (and your friend's!) devices to accelerate processing workflows with

Pando: An Easy-to-Deploy P2P Volunteer Computing Platform for the Web

Erick Lavoie, Laurie Hendren
McGill University, Montreal, Canada
erick.lavoie@mail.mcgill.ca
hendren@cs.mcgill.ca

Frédéric Desprez
INRIA Grenoble Rhône-Alpes, Grenoble, France
Frederic.Desprez@inria.fr

Motivation

Current volunteer platforms in use today (ex: BOINC platform) require a significant amount of effort and money to deploy which limits the volunteer computing approach to a few high-profile projects. Moreover, the effort in installing a client a limits the number of participants.

The web platform now offers a combination of excellent execution performance, security sandboxing, and standard communication protocol including WebRTC (peer-to-peer browser communication), and portability across many devices and operating systems, and requires no installation of client software. Moreover, social medias enable the quick mobilization of millions of users.

Those capabilities now enable the construction of newer and simpler volunteer computing platforms that may tap into more devices than ever, reach significantly more participants, and make the deployment easy for smaller and even one-shot computation projects.

Usage of Existing Hardware

Contributions

A web-based volunteer computing approach that is simple to deploy, does not require a dedicated server, and can leverage a wide variety of existing end-user devices because code is executed in browsers;

Open source implementations in JavaScript of stream abstractions, each individually available through the Node Package Manager (NPM) for reuse in other projects and that are easily composable with a growing list of community developed pull-stream modules;

An open-source command-line tool (pando), compatible with Unix pipelines that uses the stream abstractions as well as all scripts and procedures to replicate our performance experiments on the Grid'5000 testbed.

Design Goals

Clear Programming Model: Base the system on a programming model that is clear and guides a correct implementation;

Easy Deployment: Make the system easy to deploy, both for the project owner and the volunteers;

Usage of Existing Commodity Hardware: Use existing hardware including workstations, laptops, tablets and mobile phones that individuals, businesses, and universities already possess;

Scalability: Allow connecting at least a thousand volunteers;

Elasticity: Make the volunteer nodes available quickly;

Performance: Show a linear improvement in throughput compared to a single processor;

Fault-tolerance: Tolerate individual nodes that suddenly disconnect or stop answering.

Acknowledgements

NSERC CRSNG — Fonds de recherche Nature et technologies Québec — Sable McGill — McGill

Images from https://openclipart.com

Usage

1. Write or generate a JavaScript function for processing (ex: `process.js`) using the following conventions (supports NPM packages!):

```
var module.exports['/pando/1.0.0'] = function (x, cb) {
    // perform computation
    if (error) return cb(error)
    return cb(null, result)
}
```

2. Start Pando on a local laptop, desktop, or server:

```
generate-inputs | pando process.js [options] | process-outputs
```

On startup, pando provides a url at which volunteers can connect with a web browser;

3. Open the url in browser tabs on all compatible devices, one per core;

4. Share the url on social medias to invite friends to participate;

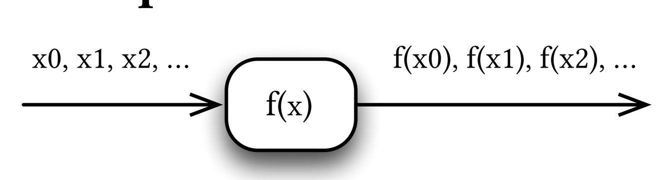
Automatic Recovery from Disconnecting Volunteers

Easy to Deploy

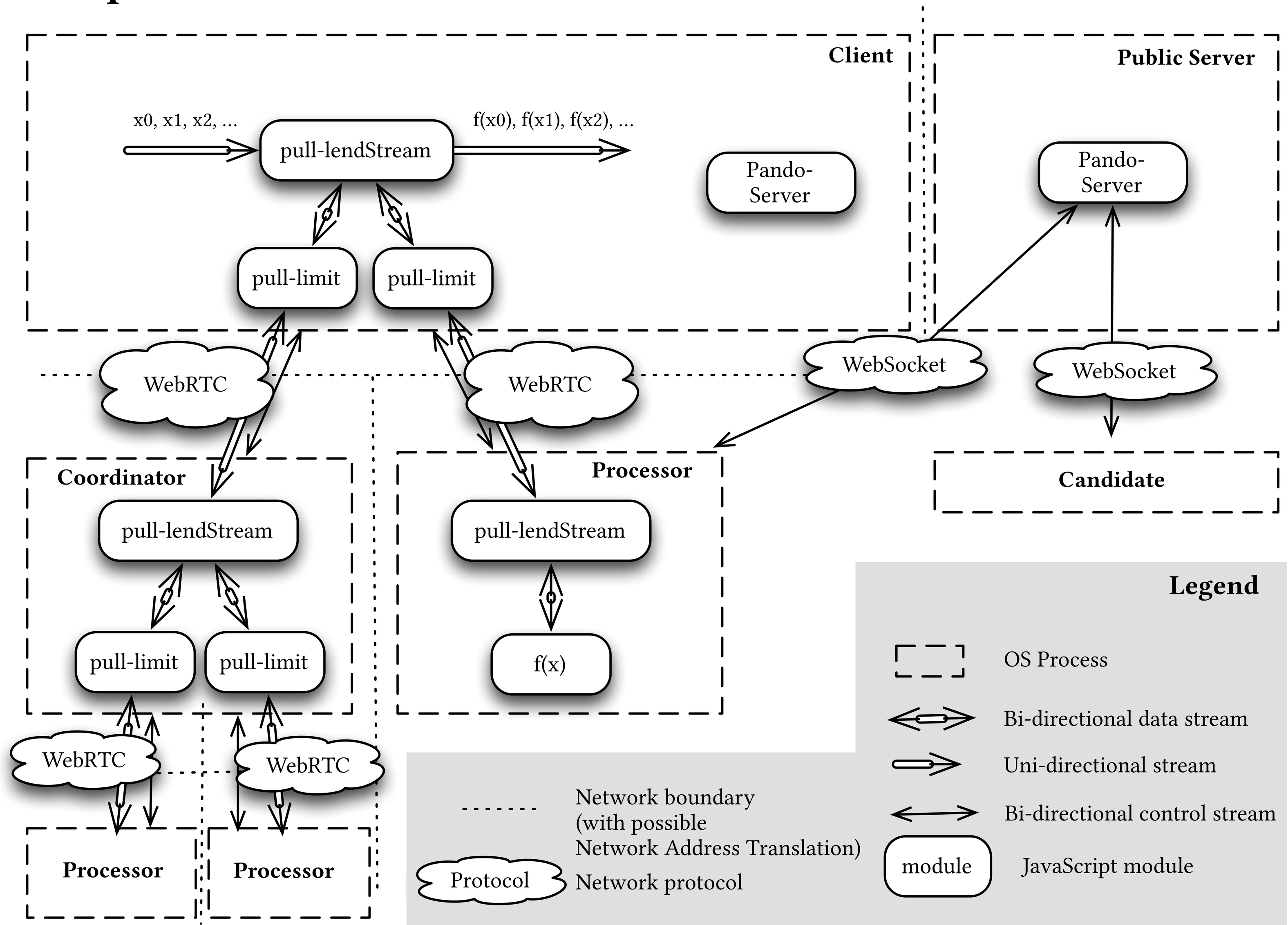
Design

Clear and Simple Programming Model

Computation Model



Components

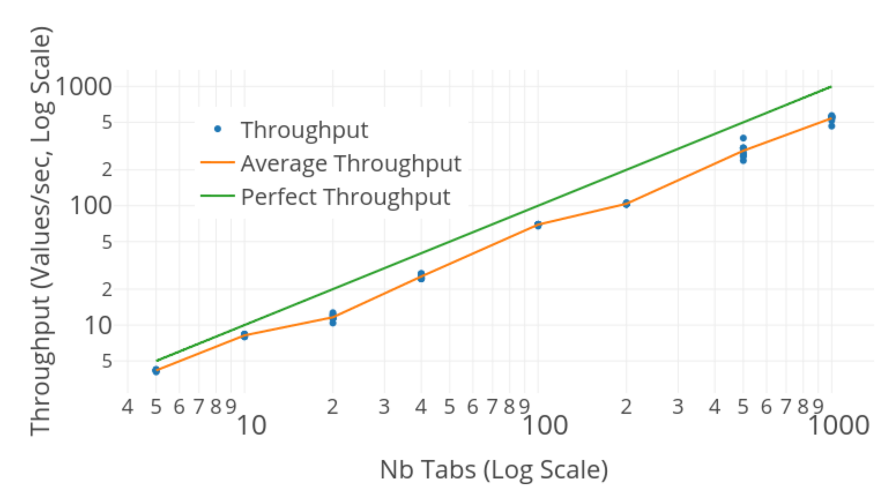


Performance

Linear speedup when increasing the number of volunteers!

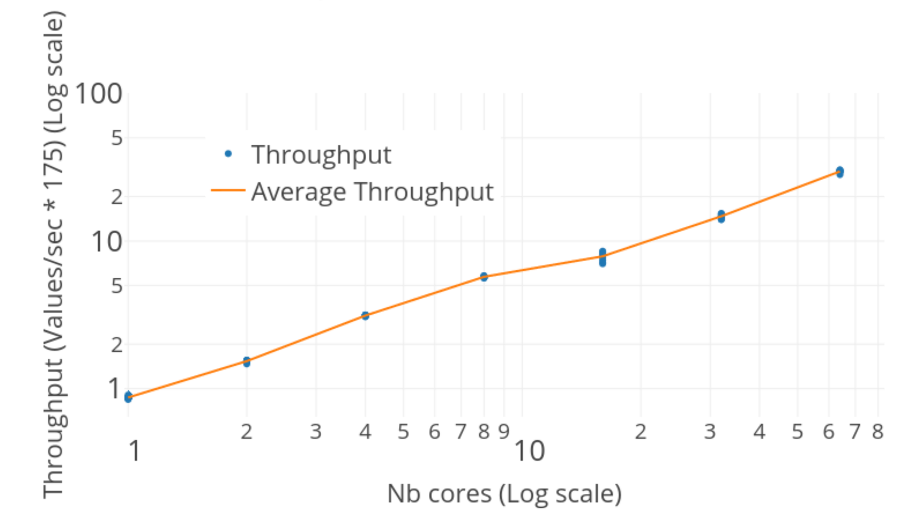
Scales up to at least a thousand browser tabs!

Squaring Number Test



* Simulated computation time (computation delayed to take 1 second), with up to 13 browser tabs/core

Collatz Conjecture



* Real computation time, 1 browser tab/core

Quickly Scalable to a 1000 Browser Tabs

Related Work

Three generations of projects, as surveyed by Fabisiak and Danilecki [1] over more than 40 articles on browser-based voluntary computing published in over 20 years, to which we add a 4th generation:

1st Generation (90s and 2000s): Based on Java and Web Applet (requires installation of plugins)

2nd Generation (2007-2010): Based on JavaScript but really slow

3rd Generation (2010-2015): Based on JavaScript but now fast! (still require server)

4th Generation (2015-...): JavaScript + Peer-to-Peer communication (no server!), browserCloud.js [2], and Pando

Conclusion and Future Work

Presented Pando, a new volunteer computing platform designed to be easy to deploy and which does not require a dedicated server and can be scaled to a thousand browser tabs for computation. This makes it useful to leverage hardware investments already made in small- and medium-businesses and university departments.

We will perform an in-depth evaluation of the performance of Pando, support more applications, optimize the performance of single computation nodes, and explore various overlay topologies for applications with more complex communication patterns during computation.

Links and References

Source Code Repository: https://github.com/elavoie/pando-computing

Demo/Teaser: https://www.youtube.com/watch?v=29ABvs3wNNI

Handbook: https://github.com/elavoie/pando-handbook

Reproduction Steps for Performance Experiments: https://github.com/elavoie/pando-handbook#publication-specific-instructions-for-reproducing-experiments

[1] Tomasz Fabisiak and Arkadiusz Danilecki. Browser-based harnessing of voluntary computing power. *Foundations of Computing and Decision Sciences*, 42(1):3–42, 2017.

[2] David Dias. browserCloud.js - A federated community cloud served by a P2P overlay network on top of the web platform. Master's thesis, Tecnico Lisboa, 2015.