Ersatz
A Discrete Event Simulation of Distributed Applications

Brett Viren

Physics Department

BROOKHAVEN
NATIONAL LABORATORY

protoDUNE DAQ Sim
8 Jul 2016

CAVEAT AUDITORIUM

This is still very much a work in progress.

Anyone interested in getting involved?
You are very welcome!

→ for now, still heavy on the concepts,
light on the results....

Caveat

The Problem

The Conceptual Model

The Software

protoDUNE/SP Data Scenarios DocDB 1086-v6

- Capture quantitative assumptions and implications and guide design.
  - (Please let's maintain **one** source for this kind of info)
- **Recently revised** (upward) by Tom Junk with help from FNAL Computing.
- 25-50Hz, 5ms, 6APA, 2-4× compression, 25-50M events.
- 25-50TB buffer disk, 30-60 parallel HDD writes, 1.5-3.0 GByte/sec.
  - Instantaneous - but taking just as many cosmics as beam between spills!

To handle this data, I wonder: **what computing and networking elements are needed, of what type, how many and how are they interconnected?**

There is a relatively high level of complexity, the assumptions have spanned at least an order of magnitude, and keep changing ⇒ need an efficient and quantitative way to explore the configuration space.

→ Simulation!

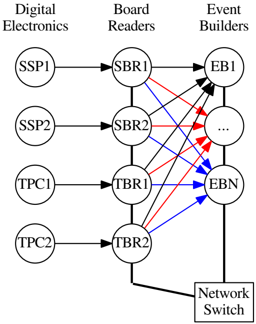
Caveat

The Problem

The Conceptual Model

The Software

Conceptual System Model



Logically, the graph is made of fully-connected layers. Physically, there are switch, NIC and computer constraints.

Model joint online/offline context as *directed acyclic graph* of *functional nodes* consuming, processing and producing *discrete units of data*.

The scope of the model may include:

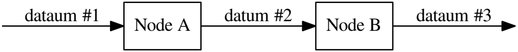
- Digital readout electronics,
- DAQ elements (ie, artDAQ nodes),
- Buffer storage units,
- Prompt processing jobs for QA/QC/commissioning.
- Networking, SATA bus.

→ **don't care about the actual content** of the data or its processing, just **data sizes, rates, processing time**, etc.

Data model

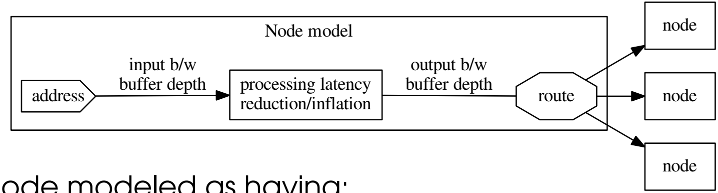
Datum[1] *a single piece of information; especially a piece of information obtained by observation or experiment; – used mostly in the plural.*
Except here, it really is singular!

- A **unit of data is discrete**, no open ended streams.
- Examples:
  - A fragment of a readout from a Board Reader
  - A single readout from an Event Builder
- A datum has effectively **just two numbers**:
  1. a size in Bytes
  2. an identifier.
- Lifetime: produced by one node, consumed by another.



[1] gcide v.0.48

Node model



Each node modeled as having:

- An **address**
- **Input bandwidth** limit and datum **buffer depth**.
- Per-datum **processing latency**.
- Data **reduction/inflation** factor.
- **Output bandwidth** limit and datum **buffer depth**.
- A **routing strategy** for addressing output datum to a downstream node.

Notes:

- Switch bandwidth still applies, but node doesn't "care".
- One node is not (necessarily) equated with one computer host.

Host model

Model a computer box to constrain nodes.

- Assert NIC RX/TX maximum bandwidth constraints.
- Restrict number of node I/O buffers (aka RAM constraint).
- Limit minimum processing time (aka CPU constraint).

I’ve not thought enough about this particular element yet....

Switch model

Model a network switch:

- Bandwidth limited at both RX and TX ports (eg, @ 1Gbps).
- Full-duplex ports, assume infinite switch fabric bandwidth.
- Shared bandwidth.
  - Invented a simple, iterative load balancing algorithm.
  - Competing transfers between shared TX and RX ports.
  - Each stream goes as fast as possible subject to fair-share.
- Preemptive.
  1. A new transfer interrupts the switch.
  2. All in-progress transfers get their progress updated.
  3. New stream added and overall bandwidth load-balanced.
  4. Continue until interrupted or next pending transfer completes.

(This has been implemented and tested.)

Discrete Event Simulation

Basic idea:

- Total system state changes in **discrete steps**.
- Changes occur based on an "**event**".
    - Most events defined in terms of a "**timeout**" primitive.

    → eg: "*do this thing in 3 seconds from now*"
- Change state by executing associated "**event callbacks**".

Example: transfer a datum on a network link:

1. Get available **bandwidth** and fixed **latency** for the link.
2. Get the **size** of the datum.
3. Set **timeout**(`now + latency + size/bandwidth`).
4. Raise event "transfer complete" and trigger associated callbacks.

→ tl;dr: focus on connecting detailed, local event callbacks and let the system work out the overall complex behavior.

Caveat

The Problem

The Conceptual Model

The Software

Ersatz

https://github.com/brettviren/ersatz

- Based on SimPy 3 and Python 3.
  - Asynchronous co-routines but single-threaded.
  - Python generators and heavy use of `yield`.
- **Ersatz** package now provides, (or will, *Real Soon Now*):
  - A shared bandwidth, preemptive network switch (done).
  - Generic, parametrized node (started).
  - Graph description and layout (todo).
  - Configuration and command line interface (started).
  - State monitoring and visualization services (rudimentary).
  - Many unit tests and examples (already)
  - Documentation (there, but lagging)

Developers welcome. Users need to wait a bit.

Watch this space.