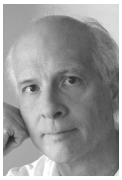
OPINION

Musings

RIK FARROW



Rik is the editor of *;login:*.
rik@usenix.org

As a certified armchair researcher, I settled into my armchair in a dark room, watching distant flashes of lightning over the desert landscape surrounding me. The scattered storms brought to mind the problem of weather forecasting, or rather, the common failings of accurate forecasts beyond the next several days.

Like other High Performance Computing (HPC) problems, weather forecasting requires enormous computational resources. The world is a big place, after all, and the weather is notoriously fickle. Other HPC problems, such as protein folding and fluid dynamics, suffer from similar issues; duplicating nature with a set of processors is an approximation, at best.

And to make matters worse, the approximations we have today must be performed on von Neumann machines.

Fish

Instead of imagining a weather system, as I sit in my armchair I imagine a school of fish. I love undersea videos of large schools of fish, light flashing off their bodies as they move in coordinated synchrony to ward off predators. These fish are not connected by networks, but use simple biological mechanisms that allow them to function as a unit—something at which our computers today are not so good.

Shifting focus a bit, I imagine dividing software into three big buckets: control, stream processing, and cells. Control software makes the decisions, such as starting other processing, and is characterized by many tests and branches. I visualize it as a tangle of threads.

Stream processing is also familiar. We now have powerful GPUs that use pipelines of specialized processors to transform a stream of data. We see stream processing in simple things, such as playing a movie or an audio recording, as well as in big data projects, where clusters of servers filter through data distributed throughout the cluster to produce the reduced result. For this, I see streams of data, with processors located along the stream like beads on a chain.

But weather forecasting, fluid dynamics, protein modeling, and the modeling of nuclear explosions do not fit into either the control or stream paradigms. In my mind, they best match cellular automata, where the state of adjacent cells affects the transition of a cell to its future state.

In weather modeling, the cells are enormous—on the scale of tens of kilometers. Worse still, the cells are three-dimensional, not at all like the classic 2D cellular

automata. And, finally, weather is affected by radiation, which will have effects not just from adjacent cells, but from the edges of the model—the surface of the earth and space itself.

As if weather is not hard enough, imagine dealing with fluid dynamics, such as modeling the turbulent flow of air across the wing of an airfoil. Here the cells are not boxes sitting on geography, but particles flowing and interacting. What each particle does affects its neighbors, and their neighbors, and so on.

Our computers are designed for control and stream processing, and do quite well at this. HPC needs to work with cells or flows or interacting particles, and include non-local effects such as radiation. Ideally, our HPC systems would work like a school of fish, rearranging themselves into the most efficient formation. Instead, we have to work with systems attached to racks, wired into networks, and arranged as a fixed grid of processors and memory. No wonder this is a hard problem to solve.

The Lineup

We open this issue with an article based on a position paper presented during HotPar '12. Rob Knauerhase, Romain Cledat, Justin Teller, and Mark Handley describe future directions for HPC systems, in particular, replacing the operating system with something much lighter weight. When one considers that supercomputers have tens of thousands of cores, running a full-fledged OS like Linux on each core would be a tremendous waste of resources—and systems such as Blue Gene already run much smaller execution engines. Knauerhase et al. explain where Intel plans to go—and is already moving—with their new Xeon Phi [1, 2] line of coprocessors.

Olivier Bonaventure, Costin Raiciu, and Mark Handley report on MPTCP, an extension to TCP that supports multiple paths without changes to client or server applications. MPTCP will be useful in datacenters, but also for mobile devices, because MPTCP is designed to *just work*—just like the IP stack we have grown accustomed to using. One huge issue in the development of MPTCP was dealing with middleboxes, which may modify header information. MPTCP neatly works around these issues, in addition to falling back transparently when a link stops working.

Kamau Wangũhũ describes VXLAN, a method for extending broadcast networks across routed networks. Virtual machines that work together may expect to be located on the same broadcast (Layer 2) network, when in reality, they may be placed wherever it is currently convenient to instantiate them. VXLAN provides transparent tunneling, so that VM instances on other networks appear to be local from the point of view of the VM.

Amandeep Khurana has written an introduction to HBase, a database that runs over Hadoop. Using HBase requires that people accustomed to using SQL databases rethink how they design their schema, and Khurana clearly provides suggestions for using HBase efficiently.

Stuart Kendrick decided that he should publish the results of his months-long study of 10 years of outages. Kendrick has the fortune of having a fairly complete record of system and software failures, and he thought that his current position demanded that he analyze this data properly. The results may not surprise you; for example, software issues lead to the most outages. If you are responsible for main-

taining many systems plus SLAs simultaneously, I suggest reading Kendrick's analysis, which does contain information useful for starting your own analysis.

Jacob Farmer managed to squeeze in an interview about a new project he and his company are working on with Harvard Medical School. The school decided to work with Cambridge Technologies to see whether they could do a better job of attaching meaning to the files that they were storing or archiving, and Farmer explains what this project means to anyone interested in making sense of the vast amounts of stored data we deal with these days.

Charles Polisher shared his and his coworkers' experience with a series of mysterious power supply failures in a datacenter. No clear cause ever emerged, but Polisher does describe the fixes that may have ended their problems.

David Blank-Edelman takes us down a different path this time. In his August '12 column, David described a tool that permits manipulating XML and HTML-structured data using paths to access that data. This time, David explores Augeas, a tool designed specifically for manipulating configuration files using paths. Like the XML Path Language, Augeas can make managing configuration files via paths much simpler than the standard pattern or field matching approaches.

David Beazley leads us down simultaneous paths in parallel by explaining how to use the multiprocessing library in Python. David first provides an example of how threads work, then shows how the multiprocessing library actually allows a Python program to use all of the cores, instead of the single core permitted by the thread library in Python. David ends his column with a simple RPC server.

Dave Josephsen exhibits his usual enthusiasm, this time for a new library called 0MQ. 0MQ abstracts away a lot of the issues with writing multicast, publish-subscribe, clients, and servers. Dave begins with a simple example of a server written using 0MQ, which actually is simpler than writing the same server without this library.

Inspired by Charles Polisher's description of failed power supplies, Robert Ferrell takes us on many more paths toward failures. For Robert, power supply failures are not sufficient, and he entertains us with other related sources of outages.

Elizabeth Zwicky continues down the path that she has been following toward more effective management, starting with two books full of ideas for dealing more effectively with team members. Then she takes a look at a book that provides the real story of the Macintosh, one without a focus on Steve Jobs, but on what actually went on in Silicon Valley garages. She then takes a hard look at a Frederick Brooks book and finishes up with a book on IPv6.

Mark Lamourine reviews two books about Coffeescript. I now have a really good idea of exactly what Coffeescript is about, and whether either or both of these books would be suitable support for learning this JavaScript replacement.

Evan Teran rounds out our reviews section with an in-depth review of *Inside Windows Debugging*. Debuggers are critical tools for developers and security geeks like Teran, who says that this book demystifies how Microsoft debuggers work.

Finally, we have summaries from the 2012 USENIX Annual Technical Conference. As always, many of the sessions at USENIX conferences are recorded, and these videos and audio recordings appear on the Web as soon as they are processed. We plan to start posting summaries online after they have been edited, which

means you can read summaries soon after an event instead of waiting until they appear in *;login:*. Summaries from the Hot Topics in Parallelism workshop and Federated Conferences Week are available online now.

The storms I've been watching have moved off to the north. From past experience—and using online weather radar—I know I can see lightning flashes further than 40 miles away. I take a deep breath as I settle deeper into my armchair.

I try to imagine a supercomputer as fluid as a school of fish, and I fail. All I can see are the racks of servers and bundles of hardwired network cables that make up the inside of today's HPC supercomputers. Even as our technology advances the von Neumann designs based on Turing's mathematics, the problems technology needs to solve require something new, something more fluid. Something I am failing to imagine.

References

[1] Knight's Corner (Xeon Phi) uses embedded Linux: http://software.intel.com/en-us/blogs/2012/06/05/knights-corner-open-source-software-stack/.

[2] Intel's Xeon Phi many-core coprocessor: http://www.anandtech.com/show/6017/intel-announces-xeon-phi-family-of-coprocessors-mic-goes-retail/.