1569613317

Bare Machine Computing Critics and Comments

The ideas of bare machine computing is intriguing and its perceived advantages are good. On one hand this whole paradigm sounds very interesting and the PIs have been demonstrating its use for several years.

The perceived benefits of BMC appear quite a bit.

Strengths:
+ Bare machine applications can potentially be useful.

The panel wants to make it clear that current proposal is not fundable but the idea has potential merit.

Again, in the end, I buy that there is value in BMC, and even value in their approach. In particular, I think there is a security argument to be made. I wish that the proposal did a better job of making it! There are situations --- the controllor for a nuclear power plant --- where you need high-level assurance. In these cases, eliminating the OS may simplify the assurance argument. I wish the proposal addressed the assurance argument directly: will they do anything with their methodology to simplify verification and certification of the AOs? VxWorks has an OS that has gone through certification. L4.verified is a microkernel with proven correctness properties. How will BMC compete?

The proposed transformation framework should be interesting, in terms of what it exposes about application/OS interactions and interfaces, and as a technique for automatically transforming that interface.

There are situations where BMC has an application and is useful.

I think there is clear intellectual merit here. The problem is interesting and challenging and has not been done before. While I am not an expert in this area, the prior work cited in the proposal seems to have to do with developing bare-machine applications from scratch, rather than systematically transforming existing OS-based applications. The approach seems sound. The PIs are obviously qualified to do this research, given all their prior work on bare-machine computing.

It has clear intellectual merit, the PIs are well qualified to do the research, and they have a clear plan for carrying out the research. However, I have a hard time believing the claimed benefits of the work. That's not to say the work doesn't have the claimed benefits, just that I am not convinced based on the presentation in the proposal.

Strengths: The most novel aspect of this proposal is the development of analysis tools to semi-automate the process of porting an application to this Bare Machine Computing system. Such a technique could potentially be useful for any of a number of lightweight kernel approaches. The technique could potentially be related to the Blue Gene system software approach, the factored OS project, or others.

The panel feels the proposed research has the potential for important results.

Overall, I think that this is a solution is search of a problem --one that cannot be tackled with the existing lightweight kernels or loaders that are already available in the scientific community.

The effort is aligned in a general way with many efforts to rethink operating systems. For example, Anant Agarwal's recent work on factored operating system (FOS) shares many of the same principles - stripping away all but the essentials, pinning user processes and system services to cores. This work could both draw on and inform work like Anant's work.

The work would increase the productivity in producing more energy efficient and secure application-specific devices.

Strengths: The main strength is that the PIs have established that this new framework is feasible through prior funded research. New OS paradigms that minimize overhead and vulnerability are potentially important to certain niche areas. Improving the development framework is an important next step.

Mark Gritter (markgritter) wrote,
2008-12-04 21:18:00

. Unapologetic Hatchet Job

Volume 36, Number 4 of "Computer Architecture News" arrived today; it was unusually thin. It consisted of 3 pages of advertisements, 7 pages of bylaws, 5 pages of comp.arch quotes, and 6 pages of an opinion piece: "Stay the course with an evolution or choose a revolution in computing" by Ramesh K. Karne, Alexander L. Wijesinha, and George H. Ford Jr. of Towson University, MD.

This was the most unintentionally amusing journal article I've yet to read.

The problem? A "myriad" of software technologies, each swiftly obsolete, consuming vast resources but providing little benefit.

In some instances, the changeover in technology has met or exceeded Moore's Law in the swiftness with which the fleeting technology has appeared only to be replaced. This has created a tremendous obsolescence in software, hardware, and people's skills. The effect on software development has been a endless requirement for additional training and retraining of personnel on the use of new tools and software environments... The long term effect is a dearth in efficiency of people's time, and capital investments and the ability to use time, people, and investments to solve new and interesting problems left unchallenged.

The solution? Get rid of the operating system.

No, really. Get rid of it. Run applications on bare hardware, as Von Neumann intended.

When all computing devices are bare, then an application, such as a Web Server, can be designed to run on any computing device.... In order to achieve this, we also believe that we need to use a common computer architecture (which could be an Intel 386, i.e. IA32, or above) as the bare machine, and a common development computing language (perhaps C++) for software creation... In such revolutionary approach, information technology development will be focused on application construction instead of developing environments. The innovation and efforts will be dedicated to development of new real-world applications such as a Web server, or some novel application to solve new software challenges, instead of focusing skills and talents to chase the latest developing environements and expending time and effort to re-host old applications.

...

So, this is not totally coming out of left field. Go read Dawson Engler's paper "Exterminate All Operating Systems Abstractions". **Karne just pushes this effort to its natural conclusion**--- *all* functionality should come from shared software (OOP) rather than an operating systems layer. And he's produced a bunch of research showing how this can be done.

What's wrong with this story?

Start by looking at all the hardware that works with Windows or Linux. Then look at the restricted Hardware Compatibility Lists for VMware ESX server or even Solaris. Bundling a few drivers into your application works fine--- Karne's students at Townson have written a few. (4?) Making your software work with every piece of hardware (even every piece of hardware that speaks IA32!) is next to impossible. Remember all those DOS games that each had to be configured individually for your sound card? Want to be first to experience the brave new world of changing all the software you own to deal with a new hardware device?

Now, vendors could (in theory) start providing C or C++ drivers--- Karne has some ideas about what should go on there. (The Linux experience says, "good luck with that.") But either the drivers all provide the same interface and make the same assumptions--- in which case we're back in "software environment" land and an obsolescence cycle--- or they don't and your application program has to deal with it. The operating system forms a useful bridge between changing hardware and existing software applications, by providing a (relatively) long-term software abstraction and a common driver model. Karne worries a lot about applications not working on new software but seems content with abandoning support for new hardware.

Which brings us to the next point. **In the Dawson/Karne world, common abstractions (files, virtual memory, TCP/IP stacks, POSIX system calls, etc.) can be provided for applications that need it via shared libraries.** Does this tend to make the problem Karne outlines better or worse? **Are we supposed to believe it is possible to halt platform churn simply by the magic of linking rather than layering? (True, it would remove the responsibility of maintaining compatibility from the end user.)**

If we did successfully halt the rat race, would it even be a good idea? I rather doubt that C++ and the IA32 architecture are the last word in computer engineering. The former is still undergoing revision and the latter is already obsolete.

Karne's answer is a sort of software socialism: an end to competitive, wasteful duplicated effort:

When this computing paradigm becomes feasible, the information technology industry will be polarized on applications. When industry establishes its focus on application development, then software productivity will reach a theoretical maximum. For any given application software effort, there would be a focus on only one development effort involved to create and enhance a given Application Object. For example, the Web server development will be done by only one development group or organization and the rest of the world can reuse it.

Successful systems research must understand not only technological issues, but market ones. In order to argue that the status quo is wrong, you first must show some understanding of why it exists. Karne blames "lack of coordination", but neither central planning nor standardized "bare hardware" is the answer. **Karne's opinion piece is not really a technical argument, it's an economic one, and bad economics at that.**

Compare this approach with virtual machines. A virtual appliance works on a variety of hardware, is insulated from upgrade difficulties, and shares resources with applications written for any operating system or even bare silicon. Arcade games from past decades are playable under emulation despite running on radically different hardware than their original packaging. The VM embraces the market's heterogeneity rather than attempting to suppress it.

Or, consider a much simpler approach to the original problem: don't waste time learning a bunch of fancy new buzzword technologies that don't offer concrete gains.
Tags: programming, rant, research, software

Killing the Computer to Save It
New York Times (10/30/12) John Markoff

SRI International computer scientist Peter G. Neumann subscribes to the philosophy that threats to computer security stem from the increasing complexity of hardware and software that has made it virtually impossible to identify system defects and vulnerabilities and ensure that they are secure and trustworthy. He argues that computers and software must be redesigned from a "clean slate." Neumann, chair of ACM's Computers and Public Policy Committee and editor of the RISKS forum, leads a research team committed to such a redesign, and he says the only practical solution is to analyze the past 50 years' research, select the best ideas, and then build a new model from the bottom up. The Clean Slate effort funds research investigating how to design less intruder-susceptible and more recoverable computer systems, with one area of focus being creating software that continuously shape-shifts to foil aspiring hackers. One design strategy Neumann's team is pursuing is tagged architecture, in which each piece of data in the system must carry encryption code credentials to guarantee the data's trustworthiness. The related capability architecture requires every software object in the system to carry special data describing its access rights on the computer, which is checked by a special element of the processor.

Complexity Is Killing IT, Say Analysts. IDG News Service (04/13/07) Krill, Paul, http://www.techworld.com/opsys/news/index.cfm?newsID=8525&pagtype=all.

- # Complexity is killing IT, say analysts
- ## Tackling the problem is getting 'harder and harder'

By Paul Krill, IDG News Service | Published: 08:05, 13 April 2007

IT complexity is getting worse, and no one has a view of the big, picture, according to technology experts at an IBM event.

Panellists from across the IT industry at IBM's Navigating Complexity conference in California painted a dire picture of IT systems taking on more and more complexity.

Harrick Vin, a vice president at outsourcing giant Tata Consultancy Services, noted that IT departments must deal with many problems, including security compliance, root cause analysis and overlapping functions.

"Unfortunately, dealing with these classes of problems is becoming harder and harder over time," said Vin, who is also a computer sciences professor at a US university. He cited a top-tier bank with more than 30,000 servers and 200,000 desktops.

The situation is compounded by the fact that different people deal with different parts of the overall problem in isolation, he said. "Essentially, what happens is we only have a silo-based understanding of what is going on."

Complexity has arisen from evolution, he added. Operating systems, applications and workload types and volumes kept changing. "The requirements that users impose onto these systems also continue to change," Vin said.

He added that systems must constantly adapt to changes. "The state of the art really is reactive firefighting," Vin said.

Peter Neumann of SRI International's computer science laboratory said old mistakes kept being repeated even if issues like buffer overflow have now been fixed.

"The problem is that we keep going through the same problems over and over and over again," Neumann said. The Multics platform had fixed the buffer overflow problem in 1965, but people ignored it, he said. Meanwhile, helpful developer tools were not being used much.

Single points of failure had presented serious problems, such as with the collapse of ARPAnet in 1980, Neumann added.

He said what was really needed was “some sort of approach to complexity that talks about sound requirements," along with good software practice.