8-1997

Cheetahs Are Fast, But Nearly Irrelevant

Calton Pu

Jonathan Walpole
Portland State University

Charles Consel

Citation Details

"Cheetahs Are Fast, But Nearly Irrelevant," Calton Pu, Jonathan Walpole, and Charles Consel, In Proceedings of the NSF Workshop on New Challenges and Directions for Systems Research, St. Louis, MO, July 31-August 1, 1997.

Title: Cheetahs Are Fast, But Nearly Irrelevant

Calton Pu, Jonathan Walpole, and Charles Consel

Dept. Computer Science and Engineering

Oregon Graduate Institute

(calton@cse.ogi.edu)

Cheetahs are among the fastest land animals on Earth, with a top speed of over 70 miles per hour. However, despite their elegant design and sheer speed, they are not a dominant species. In the African plains, they are a niche player at best, and they failed to survive in other environments. Cheetah illustrates the complexity of our echosystem, where the optimization in one single dimension (top speed) does not necessarily translate into the domination of the species in the long run, due to trade-offs in other dimensions.

Much of the past systems software research has focused on single system properties, e.g., performance as the favorite of the SIGOPS community. This narrow focus has been justified as a ``necessary simplification", with an implicit assumption that good system properties such as maintainability and reliability are somehow independent of performance, and therefore they can be added later without affecting the previous ones. As a consequence, typical systems research projects declare their ``non-goals" early and decline to investigate how their results on a single property (or a few fixed properties under restricted assumptions) would scale up and be combined with other important system properties.

While such simplifications were necessary in the past and instrumental in the construction of isolated software systems, they have revealed serious limitations in system scalability and growth. One of the most important limitations is the implicit assumption about the independence among system properties. As we have learned from the reality of large software system construction, it is difficult to build systems with multiple good system properties, including performance, reliability, and maintainability. Existing successful large software systems such as OLTP monitors illustrate the high cost of building, maintaining, and evolving such systems, by supporting the continued profitability of mainframes. The introduction of standard interface such as OLE and DCOM, explicit interface management, and interoperable interfacing tools has alleviated but not solved the problem.

One of the salient trade-offs in OLE and DCOM based systems is the system-level interoperability obtained at a high cost of inter-system communications (through remote procedure calls in this case). One way to deal with this trade-off alternates between a phase dedicated to the development of new functionality, usually in a modular fashion (DCOM style), and a performance tweaking phase, where system modules are tightened and integrated, with some sacrifice in flexibility that will add the cost and lengthen the time of further development. In an environment where fast software development is critical to the survival of the system, for example, Internet software, this trade-off demonstrates the limitations of current systems software development technology. Most of the Internet software development companies, including Netscape and Microsoft, have adopted system design choices that take this trade-off into account. However, limitations in software performance and reliability persist, particularly in in continuously evolving software systems that interact with other cooperating software systems that are evolving concurrently.

As a grand challenge to systems software research, we propose the research on systematic construction of software systems with multiple good system properties, including but not limited to:

A. Traditionally desirable ``system" properties such as performance, reliability, availability, predictability, adaptability, security, and survivability.

B. Traditionally desirable ``software engineerig" properties such as modularity, maintainability, portability, ease of evolution. C. Both well known and newly defined properties known as ``quality of service (QoS)" such as frame resolution and rate in multimedia presentations, network bandwidth guarantees, and quantitatively measurable definitions of the properties above, e.g., performance defined as ``95% of transactions with response time less than two

seconds".

D. Meta properties such as composability, decomposability, and scalability of the above properties.

Clearly, we do not imply we will build a single system with all of the above properties that will do everything under the sun. The same way many species evolved to fill the many corners in each echosystem, there will be specialized software system for each niche. The challenge is to develop the methods and tools for the construction and evolution of appropriate systems software, i.e., with the desired multiple good system properties.

There are well known technical obstacles for concrete system properties listed under items A, B, and C, that can be solved with refinements of existing technology. We see the main obstacle as the item D, on the composition and scalability of system properties. The problem is both technical and cultural. Technically, there are few established scientific and engineering approaches to system composition and continuous scalability problems faced by large software systems such as the Internet. Culturally, there is a prevailing narrowly reductionist mentality in the systems research community that considers analytical methods applied to small problems as the only scientific approach.

While the narrowly defined reductionist approach works well in traditional sciences such as biology, it has serious limitations in providing the needed meta properties such as composability and scalability. The basic reductionist assumption of property independence does not hold. Consequently, trying to improve each property in isolation is similar to using greedy algorithms to optimize a non-linear system. The usual result is getting stuck in local optima that are far from the system optimum. One example is the performance improvement of single operating system facilities by pushing work to other layers and components.

Instead of narrowly reductionist approaches, we think that successful approaches to the grand challenge posed above will result from:

1. Cooperation, integration, and composition techniques from systems research areas such as operating systems, programming languages, compilers, software engineering, and more generally from other scientific and engineering disciplines.

2. Vertical composition and integration from architecture, operating systems, libraries, middleware, up to application areas, e.g., multimedia, mobile computing, and Internet. Some examples of important research questions that fall outside of traditional areas (and therefore unfundable) include interface technology and multi-property evaluation of software systems.

3. Collaboration among researchers from different systems research areas to work together on the techniques under items 1 and 2. Typically, each area brings expertise on specific properties that are woven together by the collaboration.

Borrowing a somewhat tired buzzword, we need a paradigm shift from the current narrowly reductionist stance to collaborative, cooperative, integrative, and compositional approaches. Concretely, we need a shift at the technical level with the development of integration and composition techniques. To achieve this, we also need a shift at the cultural level making collaboration and cooperation (beyond narrowly reductionist) as reasonable, acceptable, and appropriate approaches to systems research. Otherwise, we will continue to build cheetah-style systems and write cheetah-style papers: elegant, fast, and nearly irrelevant.

Return to: Table of Contents