CPS296.2 Self-Managing Computing **January 19th, 2006**

Discussion 1: Autonomic Computing

Presenter: Shivnath Babu *Scribe: Mason F. Matthews*

1.1 Paper Summary by Shivnath Babu

The dawning of the autonomic computing era
by A. G. Ganek and T. A. Corbi

This paper is a vision/overview paper on the Autonomic Computing (AC) initiative of IBM. It begins by motivating the main motivating factors behind AC. These factors include:

1. Increasing size and complexity of enterprise systems
2. Increasing cost of administering complex systems, and the lack of a sufficient supply of trained system admins
3. Increasing time overhead for system administration
4. Lots of system outages caused by operator errors
5. Difficult to deal with changes (e.g., workload, personnel) in complex systems

Irving Wladawsky-Berger, Vice President of Strategy and Technology for the IBM Server Group, gives a nice description of AC:

"... So that instead of the technology behaving in its usual pedantic way and requiring a human being to do everything for it, it starts behaving more like the "intelligent" computer we all expect it to be, and starts taking care of its own needs. If it doesn't feel well, it does something. If someone is attacking it, the system recognizes it and deals with the attack. If it needs more computing power, it just goes and gets it, and it doesn't keep looking for human beings to step in."

The paper then discusses the four fundamental components of AC:

1. Self-configuring: The ability of a system/software to incorporate dynamic additions of features, hardware, etc.
2. Self-optimizing: The ability to monitor and tune resources automatically
3. Self-healing: The ability to quickly recover from failures/disruptions
4. Self-protecting: The ability to protect against attacks

The paper argues for an evolutionary approach to incorporate AC into the current IT infrastructure. Then it presents the importance of standards to make AC a success. The paper concludes by presenting the a summary of research at IBM on AC.

1.2 Discussion

The slides for this discussion can be found at:

http://www.cs.duke.edu/education/courses/spring06/cps296.2/lectures/lecture2.ppt

These slides presented motivating factors for AC, a definition and the basics of AC, strong and weak points of the paper, and general discussion points. Topics of discussion were as follows:

- One of the motivating factors for AC is that it is currently "Hard to deal with change." It is worth noting that this includes the difficulty of changing a system as well as the difficulty of changing a person's habits.
- As in the previous class session, the graph of time distribution for database management was controversial.
  - Shivnath noted that the Ongoing Database Administration portion (50%) included tuning the system for improved performance.
  - The graph appears to be ambiguous in a few respects. For instance, is this distribution over the lifetime of the database system, the first year of the database, or a random day drawn from the lifetime of the database system?
- We agreed that "complexity" should be more precisely defined. Some candidate metrics are:
  - Lines of code. Since bug rate can go up as lines of code increase, this may be a good starting point.
  - Number of parameters. Since the administrator needs more specialized knowledge when more parameters are present, this could be a measure of complexity. For instance, working with a parallel version of Teredata involves many more parameters, and consequently increases the install time.
  - Number or heterogeneity of devices involved in the system. This is a particularly important issue when discussing grid computing.
- When discussing the "Fundamentals of AC" slide, a few points came up:
  - Self-configuring should be performed without downtime. This includes physical configuration, adding servers, etc.
  - Index "advisors" help administrators index their queries in the proper way. They have recently been adopted by Teredata, and it is suspected that customer satisfaction has increased. Surveys on this may or may not have been conducted.
  - There are two facets of self-protecting: protecting from intruders and protecting from operators. There has been substantial work done on the first issue, but the second has been largely unexplored.

  - While IBM has developed four fundamental terms, it is worth noting that these sets of properties are not disjoint. Improving the technology in one category can have a direct impact on the others.
  - The concept of "Evolution" rather than "Revolution" has been very important to IBM.

- The slide on the paper's weak points fueled lots of discussion. For instance, students agreed that there were many goals in the paper, but not may steps. It might have been more useful to provide more examples of initial steps or components that can make current systems more autonomic.
- Examples of systems that need improvement, placement of current systems on the autonomic continuum, and implementation examples would have all been welcome additions.
- More comparisons with other work would have been useful here as well. Even a description of non-competing work would have added to this paper. Examples of similar projects are:
  - The ROC project at Stanford and Berkeley (substantial contributions by Jim Grey)
  - The Self-* project at CMU

  These projects do not have identical goals, but they are similar enough to be discussed.
- Another issue was brought up: In the motivation section of the paper, they cite a figure stating that 40% of development time is devoted to testing. Is it naive to expect AC to reduce this? Won't the addition of AC components increase the size of the testing space further?
- Is Autonomic Computing really the right solution to the problem of complexity? The paper did not address this question, but we considered a few other options:
  - Improving user interfaces may be a simpler way to attain some of the same goals. It was pointed out that this does not address the issue of recoverability, but it could reduce the number of operator errors.
  - Many systems have user groups that post configuration experiences on websites and mailing lists. A better system for searching these documents (as well as system and help files) could go a long way to reducing errors.
  - Research is being conduced on other methods for improving reliability. One such concept is Micro-Rebooting, which was studied by Trivedi and others.
- Possible steps that can move us towards AC:
  - Build systems that know themselves. Give them models representing their environments, configurations, and behavior. Allow them to have multiple configuration files and swap between them when needed.
  - Build hypothetical performance simulators so that administrators can try configurations without a risk of downtime (although this may be difficult for large systems).
  - Come up with good policies: what needs to be there to make this work? These need to be kept distinct from the AC mechanisms.
- We discussed Oracle's policies on a tangent. Their claim is that their systems are quite dynamic, and that it is possible to swap components in and out without any downtime.
- We concluded our discussion by returning to the topic of complexity. One observation was that although AC has the potential to reduce the number of system failures, it appears that those failures will involve more components and be more catastrophic. It is possible that this balance implies a point of diminishing returns.