A. Arpaci-Dusseau **Department of Computer Science**
CS739: Distributed Systems **University of Wisconsin, Madison**

Microreboot – A Technique for Cheap Recovery: OSDI'04

1 Introduction

1. What are the assumptions of this work? For providing high availability, what are the advantages of rebooting? What are the disadvantages?

2 Designing Microrebootable Software

1. How does one design micro-rebootable software (i.e., crash-only software)? Which of the five principles do you think would be hardest to follow?

3 A Microrebootable Prototype

1. The authors target large-scale Internet services. How do they modify JBoss (the open-source J2EE appliation server) to perform micro-reboots? What components can be micro-rebooted?

2. What application do they modify to be crash-only? Do you think one can derive lessons from a single application?

3. What three types of state does eBid contain? Where is each type of state stored? What is FastS and what are its pros and cons? What is SSM and what are its pros and cons? What is so important about how state is maintained in a micro-rebootable system?

4 Evaluation Framework

1. Before one can micro-reboot a component (or recover from a failure using any technique), the system must first *detect* that a failure has occurred and then *diagnose* how to fix the problem. Has previous research solved these problems? Do you think these are easy tasks?

2. What are the two approaches the authors have implemented for performing failure detection? (Note: For their evaluation, they seem to only use the second approach.)

3. What approach do the authors implement for diagnosing the failure? That is, how do they know which components need to be micro-rebooted when a failure occurs? How do they know when to stop rebooting components? (Hint: This isn't really described in Section 4.)

4. What is their *metric* for evaluating availability? Does this seem reasonable to you? What are other options for measuring availability? (You may want to take a quick look at the paper "Lessons from Giant-Scale Services" for more ideas.)

5 Evaluation Results

1. What do you think this section should be called? (My opinion: Given one application, I wouldn't call these "Evaluation Results"; I'd prefer a word more like "Demonstration". Also, they are presenting a lot of new ideas and arguments for different ways micro-reboots can be used...) Any better ideas?

2. To answer the question of "Is Microrebooting effective?" what do the authors do? Specifically, what type of faults do they inject into the application? To what values do they corrupt data? What are their conclusions? (See Table 2.)

3. When does persistent state end up getting corrupted? What is the problem with corrupting persistent state? Do you think this is acceptable?

4. To answer the question of "Is a microreboot better than a full reboot" what do they do? What do they find? Specifically, why doesn't good T_{AW} drop to 0 with microreboots? Why do some requests fail (as measured by T_{AW}) *before* the injected failure? For full process restart, why do they see some Bad T_{AW} *after* the failure? (See Figure 1.)

5. Nodes fail in large-scale clusters. Micro-reboots suggests ways to deal with these failures, but there are others. At a high level, what are the different ways the system can adjust the load when a node fails in a cluster? (Again, the "Lessons" paper might be useful.)

6. Let's examine the question of "Is microrebooting useful in clusters?" In failure-free operation, how are requests sent across nodes? When a fault occurs on one of the nodes in these experiments, how are requests redirected? What are the two primary reasons why microreboots perform better than full reboots (given FastS)?

7. What is the point of Table 5? Why doesn't this tell you the full performance cost of a micro-rebootable application? Why didn't they answer that question?

6 New Approach to Failure Management

1. In a cluster service with micro-reboots, what can be done instead of redirecting requests away from the "failed" node? How much does this second approach improve availability? What is the most extreme way of using micro-reboots with failover? Does this seem reasonable? (Section 6.1)

2. What is the idea behind microrejuvenation? Does this seem like a more or less promising ideal than microreboots? Why or why not? (Section 6.4)

7 Conclusions

1. What are the strengths and weaknesses of micro-reboots?