Difference Engine: Harnessing Memory Redundancy in Virtual Machines

General
- What is the main hypothesis underlying the paper?
- Do the authors show that the hypothesis is true?

Architecture
- Content-based sharing: Any differences from VMware approach?
- Patching: What is a patch?
  What is a reference page?
  How much memory can a patch of size X save?
- Study of sharing (Table 1): What does this table show us?
- What is a HashSimilarityDetector(k,s)? What is k? What is s?
  What does Figure 2 show us?
- How does compression work?
- What type of page needs to be identified in order to compress it?
- For compression to work effectively, what must be true about
  compression performance vs. speed of disks?
- Overall, what is the main difference between a read to a
  full-page that is shared vs. a read to a patch or compressed page?
- Why do people use the word "Architecture" instead of "Design"?

Implementation
- What is ioemu in Xen? How does it work?
- Why does ioemu make life more complicated for Diff Engine?
  How do the authors handle this problem?
- How is Clock used?
  Pages can be in states C1…C4; what are these states?
  How does DiffEngine use these states?
- How does the memory limit of Xen affect hashing of pages?
- How many different hash tables does DiffEngine use?
- Why should one be careful not to compress too many pages?
- Is there anything interesting about the swapping implementation?
  Why is it different than ESX server?

Evaluation
- Table 2: Would is shown? What would also be nice to see?
- Figure 4: What is the time-between-refs for most pages?
- Figure 5: What is being compared? What do we learn?
- Figure 8: What technique matters the most here? Why?
- Figures 9-11: What benefits does DiffEngine achieve?
  What kind of performance differences are seen between
  ESX and DiffEngine?
- What does Table 3 show? What doesn't it show?
- Figure 12: What are the authors trying to show here?
- Overall: Can aggressive swapping be used as an alternative to
  all of this machinery?