

1

GMS Design

In the GMS paper, only "clean" pages are written to global memory ("dirty" pages must be written to disk before they can participate in the GMS pool).

1.1 Why did GMS chose this restriction? What problem does it solve?

1.2 Describe how you might change GMS to safely allow it to write dirty pages to the global memory.

1.3 Why does GMS require trusted nodes? What risks would be exposed by an untrusted node?

1.4 Describe how you might change GMS to work with untrusted nodes?

Transparency vs Optimization

Butler Lampson once gave a set of principles for system design. Among these, he gave two conflicting pieces of advice on the nature of implementations. He said,

“Keep secrets of the implementation. Secrets are assumptions about an implementation that client programs are not allowed to make... Obviously, it is easier to program and modify a system if its parts make fewer assumptions about each other.”

And yet,

“One way to improve performance is to increase the number of assumptions that one part of a system makes about another; the additional assumptions often allow less work to be done, sometimes a lot less.”

That is, on the one hand we should hide an implementation for ease of development (transparency), and, on the other, we should expose our implementations for speed (optimization).

Consider this issue for each of the following three systems – Sprite, Xen, Grapevine. For each System:

- (1) Which advice of Lampson’s did the authors follow? Describe the service that was implemented, and whether the authors chose to hide or expose in the implementation.
- (2) Describe what was hidden or exposed in the implementation, and the software mechanisms that were used to do the hiding or exposing. Be specific.
- (3) Give a concrete example of how the mechanisms above were used to hide or expose in the system.
- (4) Describe one problem the authors had in utilizing their mechanism for their respective purpose, and how the authors dealt with that problem. Be specific.
- (5) Given the quotes above, discuss the authors’ goals in following the design principle they chose. Did they achieve them? Justify your answer.

2.1 Sprite

2.2 Xen

2.3 Grapevine

3 Lazy Evaluation

When implementing some service a common tradeoff is between “eager evaluation” — performing an action immediately when it is requested (hoping to amortize that cost against future accesses) and “lazy evaluation” — deferring the requested action until some later point in time (assuming that it will cost less or may not need to be done in the future). What choice to make depends both on the cost of the operation and the expected workload.

- (1) For each of the following systems, identify an important operation that uses eager or lazy evaluation as an optimization.
- (2) Explain what the operation is and what the required semantics are for it to be correct.
- (3) Describe how the implementation is eager or lazy (including any mechanisms used to ensure correctness) and why this is expected to provide an optimization in the common case.
- (4) Finally, describe a workload scenario in which this implementation will not be an optimization.

3.1 Mach

3.2 Exokernel

3.3 Xen

3.4 VAX/VMS