How the situation is new: Unlike traditional DB servers or vector supercomputers, can easily slice workload into "requests" that are isolated (in a DB sense). So can divide processing into pipeline, replicate each stage. Have a master for each request (the web server).

Key insights --- "request level parallelism" and "cluster level replication".

Software arch:
Big replicated blocks, rather than replicating the whole system, or narrow components. Feels quite different from either P2P replication, or process group model. Mostly master-slave style replication.

"End-to-end" fault tolerance; each node is not robust. RAID and other hardware fault tolerance not very useful here.

Several layers of load balancing -- at DNS, at datacenter gateway, around each replicated service.

Administration not so bad -- amortized over huge numbers, batch operations.

Hardware choices:
Because of replication, can talk about performance per unit cost. Commodity hardware, reasonably well provisioned.

Network is a shallow tree-- 100MBit ethernet links to rack switch, gigabit to centeral switch.

Microarchitecture:
Speculative processors not such a big win. Pentium IIIIs about as complex as you want.
Memory not a key bottleneck.
Power density is a major problem

Things I don't quite get: how do they know which shard to query? Try all of them?

Other work: How would you get atomic writes? Suppose you were Amazon, or gmail? Does Google's replication style still apply?

Things unspecified: Management, coordination, tracking server states.