I. **Network Services:**
   - 24x7 operation
   - huge scale (unprecedented)
   - personalization
   - no distribution problem (vs products)

Basic Advantages ofClusters:
   - Absolute scale (larger systems than any single computer)
   - High Availability -- but must tolerate partial failures
   - Commodity building blocks => cost, service and support, delivery time, alternate suppliers, trained employees

Challenges:
   - Hard to administer: single system image? ease of global view?
   - Partial failure brings new problems: must tolerate failures, can't just reboot
   - hard to have shared state (no shared address space)

ACID vs. BASE:

Idea: focus on HA with looser semantics rather than ACID semantics
   - ACID => data unavailable rather than available but inconsistent
   - BASE => data available, but could be stale, inconsistent or approximate
   - Real systems use BOTH semantics
   - Claim: BASE can lead to simpler systems and better performance (hard to prove)
      - Performance: caching and avoidance of communication and some locks (e.g. ACID requires strict locking and communication with replicas for every write and any reads without locks)
      - Simpler: soft-state leads to easy recovery and interchangable components
   - BASE fits clusters well do to partial failure and lack of a (natural) shared namespace

TACC Model:
   - Restartable Workers
      - can run anywhere (even on overflow nodes)
• Worker must handle it's own restart (easy with soft state workers, or workers that interface to an external database)
• Load balancing and worker creation/deletion is handled by SNS layer
• Fault tolerance = restart/migrate failed workers

Four kinds of workers:
• Caching: stores post-transform, post-aggregation, and WAN content
• Transformation: one-way conversion of data, including format changes (eg MIME type), resolution, size, quality, color map, language, etc.
• Aggregation: combination of data from multiple sources; eg. movie info from different theaters, company info from multiple sites (analogous to a "join" for internet content)
• Customization: support for personalization/localization based on persistent profiles

Question: is there a data independent "query" language analogous to SQL?

Starfish fault tolerance:
• idea: any alive piece can regrow (restart) the whole system
• need to track only "aliveness" not remote state (no state mirroring, since all state is soft)
• multicast to regenerate/update state (there is no difference)
• Manager watches front ends and vice versa

Burstiness and Overflow

Problem: peaks >> average => hard to plan capacity

General solutions:
• caching absorbs some spikes, especially if it can be more aggressive during overload
• admission control (especially of "hard" queries)
• overflow nodes

Burstiness is real: a side effect of humans in the loop? or just natural?

Overflow nodes:
• Idea: exploit nodes that normally have another purpose (such as desktop machines)
• Not really tried in practice so far with few exceptions, eg. Pratt & Witney run simulations on desktops at night, but not really an "overflow"
• Similar to another real world phenomenon (apocryphal?): Schwab uses managers to answer customer calls during an overflow; they are all trained but only work during overflow