1. The Vector Space Model in a Nutshell
- Documents and queries: feature vectors,
- Similarity score: cosine of enclosed angle,
- Search: compute similarity and sort results,
- Corpus matrix $C$: contains all documents,
- Compute similarities: $s = C \cdot q$ for a given query $q$,
- Partition the corpus: split $C$ row-wise or column-wise.

2. Forms of Parallelism
   - Index Replication: answer queries in parallel
   - Document Partitioning: limit search to similar clusters
   - Hybrid Partitioning: split both features and documents
   - Clustering: parallel merge-sort
   - Feature Partitioning: parallel matrix-vector product
   - Hybrid with Clustering: parallel search within clusters

3. The Memory Hierarchy
   - Index replicated
   - Index on disk
   - Document partitioned
   - Index in memory
   - Hybrid partitioned
   - Cache-friendly

4. Our Parallel Retrieval System
- Hybrid partitioning: split into equal parts,
- Dense vectors/matrices: dimensionality reduction (LSI, COV),
- Implemented using MPI: supercomputer-grade middleware,
- In-memory system: avoids slow HDDs,
- Single-precision floating point: avoids the memory bottleneck,

5. Query Response Time
   - Test Environment: 8 quad-core Xeon E5520 at 2.27 GHz, 48 GB RAM, InfiniBand network fabric, 10 Gbps,
   - Random corpus: 1024 features, and $D=10^4...10^6$ documents.
   - Document Partitioning: Hybrid Partitioning

6. Improved Response Time
- Hybrid partitioning exploits the memory hierarchy,
- Delivers super-linear speed-up over serial, in-memory system,
- Disk-based systems are not considered here,

7. Improved Throughput
- The standard for parallel search engines is index replication.
- Can a parallel program outperform multiple serial programs? Yes!
- Parallel queries/serial programs vs. serial queries/parallel program:

8. Summary
- Modern retrieval systems require dense matrix/vector algorithms,
- Exploiting the memory hierarchy is crucial for high speed-up,
- Hybrid partitioning delivers super-linear speed-up,
- Short query response time improves user satisfaction,
- Super-linear speed-up improves throughput over replication,
- MPI problematic as middleware for persistent parallel services.

9. Work in Progress
- Add clustering - conduct the parallel search within clusters,
- New middleware on top of MPI for persistent parallel services,
- Corpus analysis and feature weighting,
- Functional decomposition into components - pipelining parallelism,
- Thread-level parallelism for enhanced utilization,
- More components needed for a full search engine,
- GPGPU computing - CUDA or OpenCL numeric kernels.