Secrets of Efficient Image Search

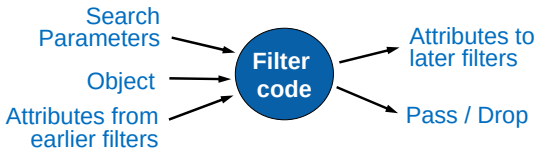
Lily Mummert, Rahul Sukthankar (Intel Research Pittsburgh)
M. Satyanarayanan (CMU), Larry Huston (Arbor Networks)

Key Ideas

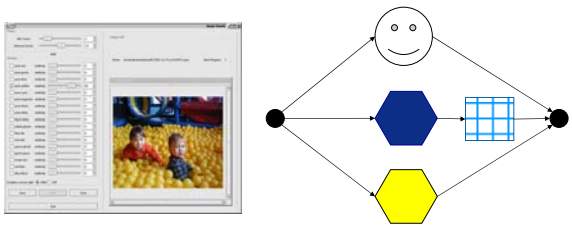
Brute-Force Search: process data without indexing
Searchlet: encapsulates domain-specific knowledge
Early Discard: eliminate hopeless data early
Active Storage: process data near storage device
Self-Tuning: dynamically adapt to avail. resources
Diamond: open source platform for interactive search

Adapt to data and query

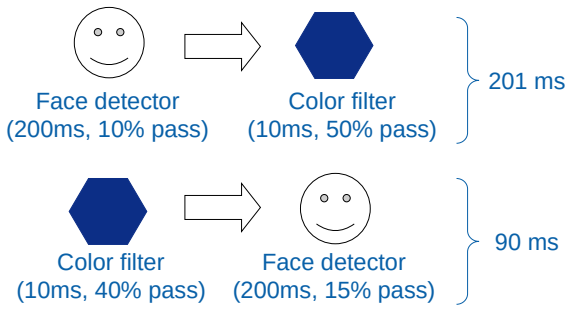
Searchlet is composed of *filters*



Filters can have dependencies
- Impose a partial order on execution
- Filters can be reordered otherwise



Filter execution order matters!
- Optimal order depends on data, query
- Goal: reject data as cheaply as possible



Diamond dynamically reorders filters
- Keeps track of filter pass rates and costs
- Periodically re-evaluates optimal filter order

Reuse previous computations

Interactive search ⇒ frequent aborts & refinements
- New search often adds one or more filters
- Old rejects remain rejects
- Idea: cache filter results for future use

Cache table for filter F

Object ID	Input Attrs	Result	Output Attrs
01	x=s1, y=s2	23	y=s3, z=s4
01	x=s9, y=s2	52	y=s3, z=s4
02	x=s7, y=s5	19	y=s6, z=s4

Diamond result cache
- Uses *memoization:* SHA-1 hash (filter, args, attrs)
- Persistent cache stores results, not object data

Attributes are worth caching too
- Problem: attributes can be larger than objects
- I/O for attribute cache interferes with object reads
- New work: determine when to save vs. recompute

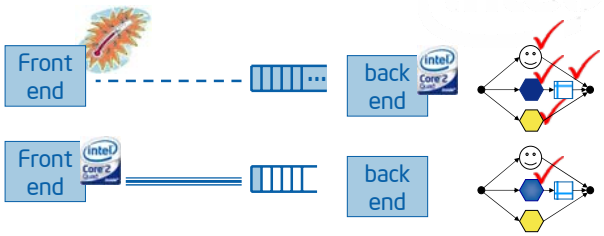
Exploit parallelism

Diamond workload is embarrassingly parallel
- Whole object, read only, any order

Diamond exploits server level parallelism
New work: exploit other sources of parallelism
- Processor: multi-core
- Storage system: multiple spindles

Adapt to hardware and workload

Diamond dynamically partitions workload
- Monitors *transmit queue length* at each stage
- Re-balances workload between front- and back-ends



Research at Intel
www.intel.com/research