Three different levels of job granularity (task, data, chunk)

If $Q$ is a subtree of $T$ then $LS_Q \subseteq LS_T$.

Common front side bus to main memory

Index Size Comparison

- Non-redundant complementing sequences
- NPS provides tree structure and $LS$ provides node labels
- A concise bijective mapping
- Simple array data structures
- Improves spatial locality and instruction level parallelism

When one or more threads is idle, jobs of smaller granularity are created

Reduce the number of operations

- Cache conscious optimizations
  - Reduce the number of operations
  - Localize the computation via simultaneous subsequence and structure matching

Index Construction

- Small tunable inverted index
- Index reduces the search space
- Exhibits very good locality
- User-defined constraints
- Easily parallelizable

Database (D) → Workload (Q) → Data Transformation → TID → Index → Query Processing Time

Query Processing Time

- Number of threads
- Time
- Time
- Time

Lcs-Trims – Tree Indexing [VLDB 07]

Bioinformatics XML Software Engineering Web mining Computational linguistics

- String matching
- Space-efficiency Sequences
- Adaptive Load-balancing Strategies

Trips & Lcs-Trims – Tree Mining [CIKM 06, 07, PVLDB 09]

Min. Support

- Frequency Counting
- Equivalence class Partitioning
- Frequent patterns

Mining Block

- On-the-Fly Embedding Lists
- chunks

Candidate Generation

- Support Counting

Frequent patterns

- Computation chunking
  - Results in constant sized memory footprints
  - Improves the locality and working set size
  - Regulates the traffic on front side bus

Trips

- Supports different job granularities

- On demand embedding list construction
  - Avoids the maintenance of persistent lists
  - Traversal between space and time

- Communication between chunks

Adaptive Parallel Strategies [PPOPP 08, PVLDB 09]

- Pattern with many matches
- Each thread is a different job

Granularity: task > data > chunk

- A multi-level work sharing methodology
  - Idle threads wait in a thread pool
  - Work is shared via job pools
  - Three different levels of job granularity (task, data, chunk)

- An adaptive approach
  - Thread pool is checked at pre-set points in the code
  - When one or more threads is idle, jobs of smaller granularity are created

Acknowledgements

This work is supported by NSF Grants CAREER-0447662, RI-CNS-0403342, NGS-CNS-0406386, and CCF-0702587.