Parallel Strategies in AI and their Applications to the Semantic Web

Jesse Weaver
Problem

• The semantic web adds new challenges to reasoning, primarily web-scale data.

• Exploiting distributed architectures is necessary for handling web-scale data.

• Parallelism is necessary to obtain decent performance on distributed architectures.
Bonacina’s Taxonomy [Bonacina 2000]

• Term-level Parallelism
• Clause-level Parallelism
• Search-level Parallelism
  – Homogeneous vs. Heterogeneous
  – Multi-Search
  – Distributed Search
Term-level Parallelism

• Fine-grained parallelism that seeks to parallelize frequent, low-level operations.
• Sometimes implemented in hardware.
• E.g., parallel unification [Gupta et al. 2001].

Unify(p(a₁,…,aₘ), q(b₁,…,bₙ))
if p=q and m=n then
  for i=1,…,n pardo
    unify(ai,bi)
  end loop
end if
End
Clause-level Parallelism

• Mid-grained parallelism that parallelizes individual inference steps.

• E.g., OR-parallelism in Prolog [Gupta et al. 2001].

  \begin{align*}
  T(?y, \text{owl:sameAs, } ?x) & : - \\
  & T(?x, \text{owl:sameAs, } ?y) .
  \end{align*}

  \begin{align*}
  T(?x, \text{owl:sameAs, } ?z) & : - \\
  & T(?x, \text{owl:sameAs, } ?y) , \\
  & T(?y, \text{owl:sameAs, } ?z) .
  \end{align*}

• E.g., backward-chained RDFS reasoning on DHTS [Kaoudi et al. 2008].
Fig. 3. BC algorithm

Figure 3 from [Kaoudi et al. 2008].
Search-level Parallelism

• Coarse-grained parallelism that seeks to divide the search space among processes.

• Each process operates on its own data.

• Involves communication between processes.
Search Strategy

• A search strategy $\mathbf{C}$ is a pair $<\mathbf{I}, \mathbf{S}>$ where $\mathbf{I}$ is an inference system and $\mathbf{S}$ is a search plan.

• $\mathbf{I}$ provides the rules for deduction, whereas $\mathbf{S}$ provides a concrete method of applying the rules.

• In other words, $\mathbf{S}$ decides how to deal with non-determinism in the inferencing process.
Homogeneous vs Heterogeneous

• In homogeneous inference systems, each process uses the same inference system.

• In heterogeneous inference systems, each process uses a different inference system.

• E.g. heterogeneous, Approximate, Anytime Reasoning by composition [Rudolph et al. 2008].
Figure 1 from [Rudolph et al. 2008].

**Fig. 1.** Defect over time.
Multi-search

• Each process employs a different search plan.
• Forms of non-determinism:
  – Don’t-know non-determinism: each path from a choice-point may or may not lead to a solution.
  – Don’t-care non-determinism: all paths from a choice-point lead to a solution.
• Parallelize “don’t-know” = parallel searches
• Parallelize “don’t-care” = optimized search
Distributed Search

• Seeks to derive parallelism by subdividing the search space at every inference step.

• Basically boils down to data partitioning in parallel programming.

• The most common approach.
Distributed Search in Semantic Web

- [Guo et al. 2006] Partition Abox wrt Tbox to create independent partitions for **OWL Lite** reasoning.
- [Soma et al. 2008] Graph, Hash-based, and Domain-specific data partitioning. Min-cut rule partitioning. **OWL Horst**.
- [Weaver et al. 2009] Replicate ontology to all processes for independent, **near-complete RDFS** reasoning.
- [Urbani et al. 2009] MapReduce system that replicates ontology to all reducers for **RDFS Munoz+ (plus CMPs and DTs from RDFS)** reasoning.
- [Urbani et al. 2010] WebPIE MapReduce system for **OWL Horst** reasoning.
- [Goodman et al. 2010] Cray XMT used for **RDFS Munoz** reasoning.
- [Hogan et al. 2010] SAOR uses ontology to instantiate rules, broadcasts rules to all processes, and then performs **partial OWL 2 RL** reasoning independently, in parallel.
## Comparison

<table>
<thead>
<tr>
<th>System</th>
<th>Reasoning</th>
<th>Dataset</th>
<th>NP</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Guo et al. 2006]</td>
<td>Complete OWL Lite</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[Soma et al. 2008]</td>
<td>Complete OWL Horst</td>
<td>LUBM-10 (&gt; 10M)</td>
<td>16</td>
<td>-</td>
</tr>
<tr>
<td>MaRVIN [Oren et al. 2009]</td>
<td>Eventually Complete RDFS</td>
<td>SwetoDBLP (15M)</td>
<td>64</td>
<td>23m</td>
</tr>
<tr>
<td>[Weaver et al. 2009]</td>
<td>Near Complete Finite RDFS</td>
<td>LUBM(10k,0)/4 (345M)</td>
<td>128</td>
<td>8.5m</td>
</tr>
<tr>
<td>[Urbani et al. 2009]</td>
<td>Complete RDFS Munoz+</td>
<td>DBpedia (150M)</td>
<td>combined sets (30B)</td>
<td>32</td>
</tr>
<tr>
<td>“Speeddating” [Kotoulas et al. 2010]</td>
<td>Complete RDFS Munoz</td>
<td>SwetoDBLP (15M)</td>
<td>combined sets (195M)</td>
<td>64</td>
</tr>
<tr>
<td>WebPIE [Urbani et al. 2010]</td>
<td>“Complete” OWL Horst</td>
<td>Uniprot (1.5B)</td>
<td>LUBM? (102B)</td>
<td>32</td>
</tr>
<tr>
<td>Cray XMT [Goodman et al. 2010]</td>
<td>Complete RDFS Munoz</td>
<td>LUBM(40k,0) (5.34B)</td>
<td>512</td>
<td>1.75m</td>
</tr>
<tr>
<td>SAOR [Hogan et al. 2010]</td>
<td>Partial OWL 2 RL</td>
<td>crawled (1.12B)</td>
<td>8</td>
<td>3.33h</td>
</tr>
</tbody>
</table>
References


Questions?