Parallelism in Logic Programs

Vítor Santos Costa

COPPE/Sistemas and LIACC

UFRJ and UP





Logic Programming

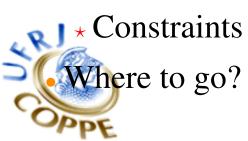
- Declarative Approach to Programming
- Prolog:
 - ⋆ Practical
 - ⋆ Popular
 - * Hard to Change
- Can Parallelism help?





Outline

- Quick Review of Parallelism
 - ★ Explicit
 - ⋆ Implicit
- Applications
 - ***** Deterministic
 - ***** Model Checking
 - ★ ILP





Parallelism in LP

- Explicit:
 - ★ MPI
 - ⋆ Threads
 - Programming Languages
- Implicit
 - ⋆ Or-Parallelism
 - Dependent And-Parallelism
- Independent And-Parallelism



MPI Packages

- Distributed Model
- Low-Level Interface:
 - ★mpi_
- Distributed DB
- Issues
 - ***** Imperative
- Message Passing Expensive?



Threads

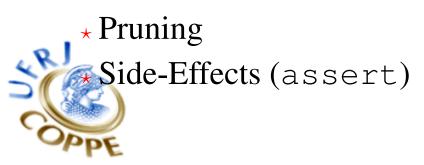
- Thread hosts *engine*
- Shared Data-Base:
 - Private engine predicates
- Implementation
 - ***** P-Threads
 - DataBase Concurrency Management
- Issues:

* Maintenance No standard (ciao, SWI, SICStus). Explicit Locking



OR (Search) Parallelism

- Usually, Multi-Agent Model
- Implementation well studied:
 - Minor (?) Engine changes
 * Low space-time overhead
 * Scheduling
- Issues:





Independent AND-Parallelism

- Divide-And-Conquer
- Works very well for deterministic computations
- Implementation:
 - ★ Goal Setting-Up
 - Minor Engine Changes
- Issues:
 - * Backtracking
- Memory Fragmentation Detecting Fork Points



Dependent AND-Parallelism

- Concurrent Computation
- Usually, deterministic model
- Implementation:
 - ★ Goal Setting-Up
 - Major Engine Changes
 - ⋆ Concurrent GC
- Issues:
- Language Issues



Status?

- Great tech:
 - ★ Smart Work
 - ⋆ Cool Speedups
- But, Low Impact
 - * Too focused on small benchmarks
 - ★ Everest Effect
 - * Runs well for

* Too few apps, on * Too few machines Hard To Maintain



Can Parallelism Help?

- Who can it help?
 - ***** Application
- Where can it help?
 - ⋆ Hardware
- How can it help?
 - ★ Explicit, ORP, IAP?
- •The Answer: •Applications are the key!!



Deterministic Applications

- Very Many Examples:
 - ***** Compilers in Prolog
 - van Noord's Finite State Automaton
 - * Angelopoulos's Markov Chain Monte Carlov
- Patterns:
 - * Memory Management is Crucial (GC)

Small Procedures may dominate running time



So Far

- ORP is useless here
- IAP has had good results
 - ★ Compilers
- DAP can also work well
- Problem: Memory Management
 - * Work-Set can be pretty large
- Incremental Parallel GC?



Search Applications

- Backtracking Search is Core
- Search Space is an Issue:
 - ⋆ Tabling
 - Co-routining/Constraint Propagation
- Search can be improved
 - * Meta-Interpreter (ILP)





Search Applications

Issues

- Memory Management:
 - * Often, search space is represented in DB:
 - * Explicitely
 - * Through tables
 - Execution Stacks ok
- Same engine may run very different searches

Harder to understand performance



Example I: Model Checking

- XMC:
 - * From several spec languages
 - * To logic programs run by Tabled Prolog
- Tabling avoid loops, guarantees finiteness
- Models be deterministic
 - * or may be shallow search with very high branching factor
- ∧ or may be deep search
 - Related: Program Analysis, Security



So Far

• ORP can work well:

- * Excellent results of OPTYap
 - * Best case, linear up to 32
- * No Side-Effects
- ***** Extensive Search
- Lots of Interest in Parallelism
- Work going on on Threads



Example II: Learning with ILP

- Search Engine generates *clauses*
- Clauses are evaluated on examples
- Performance?
 - ***** Time may be spent on search ops
 - * or on running examples:
 - * We may have *lots* of examples
 - * Each example may be costly



So Far

- Gobs of Interest
- Grid used for experiment management
- MPI with new ILP search techniques
- Threads:
 - * Randomised Search is easy to parallelise
 - * Clause scoring: improving

Exciting opportunity for Implicit Parallelism!



Example III: Constraints

- Two phases
 - ***** Constraint Propagation
 - * Labeling
- How does it run?
 - Labeling is search
 - * Propagation is deterministi
- Which matters? Depends on data...



So Far

- ORP based
 - ***** Good Results
 - * Not widely used
- DAP
 - ***** Parallelise Constraint Propagation
 - * Interesting Results in Andorra-I
 - Related to Distributed Constraints





Conclusions

- Speed is a Real Issue:
 - * Apps can run for hours
 - * Parallelism is Useful
- Often, Memory Intensive (Stacks or/and DB)
- Same program performs different on different data
 - ⋆ Flexibility
- Low Overhead



Applications

- Deterministic Applications
 - * Not well exploited
 - ***** GC is an issue
- Search
 - * ORP can do well
 - * Lots of Interest: ILP, MC





Future Directions

• Explicit Parallelism is having Real Impact:

★ eg, best paper at this year's ILP

- Implicit Parallelism
 - ★ Needs to Fit
 - ⋆ ORP
 - * Does well for pure search
 - * Benefits from Threads
 - ★ IAP

* Lots of potential for deterministic comps Work on Memory Management



SMTs

- Shared Memory is Great!
- Applications/Interest exists
 - * Limited Speedup is Speedup!!
- But:
 - * Memory Bandwith?
 - ⋆ Maintenance?

High-Performance Context-Switching?



Fine Grained Parallelism

- Back?
- Ex: Concurrent GC
- Ex: WAM-level
 - New Compilation Technology
 - * Real Exciting
- Bad experience with Hyperthreading
- Better Thread Packages?

