Speculative Plan Execution for Information Agents

Greg Barish
University of Southern California
June 30th, 2003

Thesis Committee
Prof. Craig Knoblock (chair)
Dr. Steven Minton, Fetch Technologies
Prof. Paul Rosenbloom
Prof. Cyrus Shahabi
Prof. Jean-Luc Gaudiot (external member)
Outline

1. Introduction and motivating example

2. Thesis statement & contributions

3. Expressive & efficient information agent plans

4. Speculative plan execution

5. Value prediction for speculative execution

6. Related work

7. Summary & future work
Information agents

• Automate the querying of data networks (e.g., the Web)
  – Gather, combine & process data from multiple remote sources (e.g., Web sites)

• Sample information agent task:
  – **Buying a used car**: safety ratings and reviews for certain criteria
  – **Example**:
    • 2002 Midsize coupe/hatchbk, $4K-$12K, no Oldsmobiles
1. Locate cars that meet criteria - Edmunds.com

2. Filter out Oldsmobiles
The CarInfo agent

1. Locate cars that meet criteria
   - Edmunds.com

2. Filter out Oldsmobiles

3. Gather safety reviews for each
   - NHSTA.gov
The CarInfo agent

1. Locate cars that meet criteria
   - Edmunds.com

2. Filter out Oldsmobiles

3. Gather safety reviews for each
   - NHSTA.gov

4. Gather detailed reviews of each
   - ConsumerGuide.com
ConsumerGuide navigation

New Car Pricing & Reviews
2002 Dodge Stratus

Highlights for 2002

Stratus sedans share a design with the Chrysler Sebring sedan and convertible. Stratus coupes share a design with the Chrysler Sebring coupe.

Sedans come in SE, SXT, SE Plus, ES, and new R/T trim. The SXT and both SE versions come with a 4-cyl engine and offer an optional Chrysler-made 2.7-liter V6. The V6 is standard on the ES and R/T. All but the R/T have mandatory automatic transmission. All sedans have 4-wheel disc brakes, with ABS optional. Curtain side airbags are optional; no torso side airbags are offered. Added at midyear, the R/T sedan has antilock 4-wheel disc brakes, a 5-speed manual transmission, and offers at no extra charge Chrysler’s AutoStick automatic transmission with manual shift gate.

Coupes use powertrains and platforms from Mitsubishi’s Eclipse and Galant. They come in SE and R/T models. The SE has a 4-cyl engine or optional 3.0-liter V6. The V6 is standard on the R/T. Both coupes use manual transmission or optional automatic. R/T automatics come with traction control and can be ordered with AutoStick. Four-wheel disc brakes are included with the V6. Among coupes, ABS is optional only on the R/T.

Competition Perennial Best Buys Honda Accord and Toyota Camry continue to shine with refinement, model diversity, and comfort. Both come in coupe and sedan forms, offer economic 4-cylinder or sporty V6 power, have room for four adults, and are reasonably priced.
Agent Execution Performance

- **Standard von Neumann model**
  - Execute one operation at a time
  - Each operation processes all of its input before output is used for next operation
  - **Assume:** 1000ms per I/O op, 100ms per CPU op

- **Execution time = 13.4 sec**
Streaming dataflow model

• **Dataflow**
  – Operations scheduled by data availability
    • Independent operations execute in parallel
    • Maximizes *horizontal parallelism*
  – Dataflow computers [Dennis 1974] [Arvind 1978]
  – Example: computing \((a*b) + (c*d)\)

• **Streaming**
  – Operations emit data as soon as possible
    • Independent data processed in parallel
    • Maximizes *vertical parallelism*
  – Network query engines
    [Ives et al. 1999] [Naughton et al. 2000] [Hellerstein et al. 2001]
Dataflow-style CarInfo agent plan

(Midsize coupe/hatchback, $4000 to $12000, 2002)

((Dodge Stratus),
(Pontiac Grand Am),
(Mercury Cougar))

WRAPPER ConsumerGuide Search

WRAPPER ConsumerGuide Summary

WRAPPER ConsumerGuide Full Review

WRAPPER NHTSA Search

JOIN

(search criteria)

WRAPPER Edmunds Search

SELECT maker != "Oldsmobile"

((Oldsmobile Olero),
(Dodge Stratus),
(Pontiac Grand Am),
(Mercury Cougar))

((http://cg.com/summ/20812.htm),
other summary review URLs)

((http://cg.com/full/20812.htm),
other full review URLs)
Streaming dataflow performance

- Improved, but plan remains I/O-bound (76%)
- **Main problem**: remote source latencies
  - Meanwhile, local resources are wasted
- **Complicating factor**: binding constraints
  - Remote queries dependent on other remote queries

- **Question**: How can execution be more efficient?
Speculative execution of streaming dataflow plans increases the degree of run-time parallelism for information agents.
Speculative plan execution

- Execute operators ahead of schedule
  - Predict data based on past execution

- Allows greater degree of parallelism
  - Solves the problem caused by binding constraints
Contributions of thesis

• **Expressive plan language & efficient execution system for information agents**
  – *Dataflow plan language* that enables more than basic querying
  – *Thread-pool model* of streaming dataflow execution

• **An approach to speculative plan execution**
  – *Safe & fair*
  – Yields arbitrary speedups
  – Algorithm for the *automatic transformation* of agent plans

• **An approach to value prediction**
  – Combines *caching, classification*, and *transduction*
  – Better accuracy and space efficiency than strictly caching
Outline

1. Introduction and motivating example
2. Thesis statement & contributions
3. Expressive & efficient information agent plans
4. Speculative plan execution
5. Value prediction for speculative execution
6. Related work
7. Summary & future work
Expressive agent plan language

• Operators support:
  – Web data gathering
  – Data manipulation
  and...
  – Conditional execution
  – Monitoring
  – Async communication
  – Agent management
  – Extensibility

• Subplans
  – Modularity, reusability
  – Recursive subplans
Expressing the CarInfo agent plan

PLAN car-info {
    INPUT: criteria
    OUTPUT: reviews-and-ratings
    BODY {
        Wrapper ("Edmunds", criteria : cars)
        Select (cars, "maker != 'Oldsmobile'": filtered-cars)
        Wrapper ("NHTSA", filtered-cars : safety-ratings)
        Wrapper ("CG Search", filtered-cars : summary-urls)
        Wrapper ("CG Summary", summary-urls : full-urls)
        Wrapper ("CG Full", full-urls : car-reviews)
        Join (safety-ratings, car-reviews, "l.make=r.make and l.model=r.model": reviews-and-ratings)
    }
}
Streaming dataflow executor

- Thread pool architecture
  - Enables dynamic parallelism without losing control

Example:

**Thread Pool**

**Plan operators**
(e.g., Wrapper, Select, etc.)

**Plan Input**

**Plan Output**

Example:

- Wrapper: Edmunds Search
- Select: maker != "Oldsmobile"

((Oldsmobile Olero),
(Dodge Stratus),
(Pontiac Grand Am),
(Mercury Cougar))

(Midsize cpe/hatchbk,
$4000 to $12000,
2002)
Experimental results

- **Hypothesis #1**
  - Language and executor enable efficient information agents

- **Hypothesis #2:**
  - Language is more expressive than query languages of other network query engines

- **Hypothesis #3:**
  - Added expressivity does not detract from performance
Outline

1. Introduction and motivating example
2. Thesis statement & contributions
3. Expressive & efficient information agent plans
4. Speculative plan execution
5. Value prediction for speculative execution
6. Related work
7. Summary & future work
How to speculate?

• General problem
  – Means for issuing and confirming predictions

• Two new operators
  – Speculate: Makes predictions based on "hints"
  – Confirm: Prevents errant results from exiting plan
How to speculate?

- **Example: CarInfo**
  - Make predictions about cars based on search criteria
  - Makes practical sense:
    - Same criteria will always yield same cars
How to speculate?

• Example: CarInfo
  – Make predictions about cars based on search criteria
  – Makes practical sense:
    • Same criteria will always yield same cars
Detailed example

<table>
<thead>
<tr>
<th>2002</th>
<th>Midsize coupe</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$4000-$12000</td>
</tr>
</tbody>
</table>

Time = 0.0 sec
Issuing predictions

<table>
<thead>
<tr>
<th></th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oldsmobile Olero</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dodge Stratus</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pontiac Grand Am</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mercury Cougar</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Time = 0.1 sec
Speculative parallelism

Time = 0.2 sec
Answers to hints

Time = 1.0 sec
Continued processing

Additions (corrections), if any

T1 T2 T3 T4

W Speculate S W W J W Confirm

Time = 1.1 sec
Generation of final results

Dodge Stratus (safety) (review) T2
Pontiac Grand Am (safety) (review) T3
Mercury Cougar (safety) (review) T4

Time = 4.2 sec
Confirmation of results

Dodge Stratus (safety) (review)
Pontiac Grand Am (safety) (review)
Mercury Cougar (safety) (review)

Time = 4.3 sec
Safety and fairness

• **Safety**
  – Confirm blocks predictions (and results of) from exiting plan before verification

• **Fairness**
  – **CPU**
    • Speculative operations executed by "speculative threads"
      – Lower priority threads
  – **Memory and bandwidth**
    • Speculative operations allocate "speculative resources"
      – Drawn from "speculative pool" of memory
      – Other solutions exist, such as RSVP (Zhang et al 1994)
Getting better speedups

• Cascading speculation
  – Single speculation allows a max speedup of 2
    • Time spent either speculating or confirming
  – Cascading speculation allows arbitrary speedups
    • Up to the length of the longest plan flow
Cascading speculation in CarInfo

- Use predicted cars to speculate about the ConsumerGuide summary and full URLs

- Optimistic performance
  - Execution time: \( \text{max} \{1.2, 1.4, 1.5, 1.6\} = 1.6 \text{ sec} \)
  - Speedup over streaming dataflow: \( \frac{4.2}{1.6} = 2.63 \)
Automatic plan transformation

- **Amdahl's Law:**
  - Focus on most expensive path (MEP)

- **Basic algorithm**
  1. Find MEP
  2. Find best candidate speculative plan transformation
  3. IF no candidate found, THEN exit
  4. Transform plan accordingly
  5. REPEAT

- **The "best" candidate**
  - The one with the highest potential speedup

- **Algorithm assumes some addtl speculative overhead**
  - Function of the amount of data speculated about
Web agent experiments

Average speedup (first tuple)

Average speedup (last tuple)
Distributed database experiments

- **TPC-H benchmark**
  - Adhoc business queries for an order-entry schema
  - Modeled each entity (table) in the schema as a remote source

- **Experiment**
  - Varied latency and database scale
  - Tested on recurring queries

![Graph showing speedup for TPC-H queries with varying latencies]

- Theoretical max speedup for each TPC-H query (Q2 to Q20) with latency levels of 2000ms, 4000ms, 6000ms, 8000ms, and 10000ms.
Outline

1. Introduction and motivating example
2. Thesis statement & contributions
3. Expressive & efficient information agent plans
4. Speculative plan execution
5. Value prediction for speculative execution
6. Related work
7. Summary & future work
Value prediction

- Better value prediction = better speedups

- Prediction capability

<table>
<thead>
<tr>
<th>Category</th>
<th>Hint</th>
<th>Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Previously seen</td>
<td>Previously seen</td>
</tr>
<tr>
<td>B</td>
<td>Never seen</td>
<td>Previously seen</td>
</tr>
<tr>
<td>C</td>
<td>Never seen</td>
<td>Never seen</td>
</tr>
</tbody>
</table>

- Examples:

Edmunds car list from search criteria

H → 2002 Midsize coupe 4K-12K

P → Olds Olero, Dodge Stratus, Pontiac Grand Am, Mercury Cougar

ConsumerGuide full review URL from summary URL

http://cg.com/summary/20812.htm
http://cg.com/full/20812.htm

http://cg.com/summary/12345.htm ?
http://cg.com/summary/12345.htm ?
Value prediction techniques

• Caching
  – Associate a hint with a predicted value

• Classification
  – Use features of a hint to predict value
  – EXAMPLE: Predicting car list from Edmunds

<table>
<thead>
<tr>
<th>Year</th>
<th>Type</th>
<th>Min</th>
<th>Max</th>
<th>Car list</th>
</tr>
</thead>
<tbody>
<tr>
<td>2002</td>
<td>Midsize</td>
<td>8000</td>
<td>15000</td>
<td>(Oldmobile Olero, Dodge Stratus)</td>
</tr>
<tr>
<td>2002</td>
<td>Midsize</td>
<td>7500</td>
<td>14500</td>
<td>(Oldmobile Olero, Dodge Stratus)</td>
</tr>
<tr>
<td>2002</td>
<td>SUV</td>
<td>14000</td>
<td>20000</td>
<td>(Nissan Pathfinder, Ford Explorer)</td>
</tr>
<tr>
<td>2001</td>
<td>Midsize</td>
<td>11000</td>
<td>18000</td>
<td>(Honda Accord, Toyota Camry)</td>
</tr>
<tr>
<td>2002</td>
<td>SUV</td>
<td>18000</td>
<td>22000</td>
<td>(Nissan Pathfinder, Ford Explorer)</td>
</tr>
</tbody>
</table>

Decision list

- type = SUV: (Nissan Pathfinder, Ford Explorer)
- type = Midsize
  - min <= 10000: (Olds Olero, Dodge Stratus)
  - min > 10000: (Honda Accord, Toyota Camry)
Value prediction techniques (cont'd)

• Transduction
  – Transducers are FSA that translate hint into prediction

To create full review URL:
1. Insert "http://cg.com/full/
2. Extract & insert the dynamic part of the summary URL (e.g., 20812)
3. Insert ".htm"

```
http://cg.com/summary/20812.htm
```

```
http://cg.com/full/20812.htm
```
Value transducers

• Synthesize predictions from hints

• Identify predicted value "templates"
  – Alternating seq of STATIC/DYNAMIC elements

• Value transducers built from templates
  – State transitions (arcs) = high-level operations:
    • INSERT, CACHE, CLASSIFY, TRANSDUCE (hint chars)
Learning value transducers

• Identify **STATIC/DYNAMIC** template
  – LCS-based approach (Hirschberg 1975) to identify answer template

• For each **STATIC** element,
  – Construct **INSERT** arc to next automata state

• For each **DYNAMIC** element,
  – Construct **TRANSDUCE**, **CLASSIFY**, or **CACHE** arc to next automata state
  • Inducing character-level hint transducer also requires identifying a template -- from the hints
Detailed example: CarInfo URLs

**HINTS:**

**ANSWERS:**
Experimental results

- More space efficient than strictly caching

**Hint classification**
(CarInfo summary review URL)

**Hint transduction**
(CarInfo full review URL)

![Space savings (over caching)](chart)

<table>
<thead>
<tr>
<th>Number of examples</th>
<th>0.00%</th>
<th>20.00%</th>
<th>40.00%</th>
<th>60.00%</th>
<th>80.00%</th>
<th>100.00%</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>400</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>600</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>800</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Number of examples](chart)

<table>
<thead>
<tr>
<th>Number of examples</th>
<th>2</th>
<th>10</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20.00%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40.00%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60.00%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80.00%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100.00%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Experimental results

- Better accuracy than strictly caching

**Hint classification**

![Graph showing accuracy vs. number of new examples for different predictors.]

<table>
<thead>
<tr>
<th>Predictor</th>
<th>Average number of examples required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car-Full</td>
<td>3</td>
</tr>
<tr>
<td>Rep-Graph</td>
<td>8</td>
</tr>
<tr>
<td>Phone-Detail</td>
<td>3</td>
</tr>
</tbody>
</table>
Outline

1. Introduction and motivating example
2. Thesis statement & contributions
3. Expressive & efficient information agent plans
4. Speculative plan execution
5. Value prediction for speculative execution
6. Related work
7. Summary & future work
Related Work

• **Efficient agent execution**
  – **Dataflow computers** [Dennis 1974] [Arvind et al. 1978]
    • Parallel programming languages (Val, Id, SISAL, Haskell)
    • Languages for embedded systems (Verilog, VHDL)
  – **Network query engines**
    • Tukwila [Ives et al. 1999] Niagara [Naughton et al. 2001]
      Telegraph [Hellerstein et al. 2001]
  – **More general agent executors**
    • RPL [McDermott 1991], RAPs [Firby 1994], PRS-Lite [Myers et al. 1996]

• **Speculative execution**
  – **Approximate & partial query results** [Hellerstein et al. 1997]
    [Shanmugasundaram et al. 2000] [Raman and Hellerstein 2001]
  – **Executing anticipated actions in advance**
    • Continual computation [Horvitz 2001], time-critical decision making
      [Greenwald and Dean 1994]
Related Work

• **Speculative execution (cont'd)**
  – **Predicting commands**
    • Command line prediction [Davison and Hirsh 2001], assisted browsing [Lieberman 1995]
  – **Other types of speculative execution**
    • File system prefetching [Chang and Gibson 1999], control speculation in workflow processing [Hull et al. 2000]
  – **Network prefetching**

• **Learning value predictors**
  – **Value prediction as speedup learning** [Fikes et al. 1972], [Mitchell 1983], [Minton 1988]
  – **Transducer learning** [Oncina et al. 1994] [Hsu and Chang 2001]
  – **URL prediction** [Zuckerman et al. 1999] [Su et al. 2000]
Outline

1. Introduction and motivating example
2. Thesis statement & contributions
3. Expressive & efficient information agent plans
4. Speculative plan execution
5. Value prediction for speculative execution
6. Related work
7. Summary & future work
Summary of contributions

• An expressive language and efficient execution system for information agents

• An approach to speculative execution of information agent plans
  – Can yield arbitrary speedups
  – Safe, fair

• Value prediction approach that combines caching, classification, and transduction
  – More accurate & space efficient than strictly caching
Future work

• Learning to compute speculative overhead

• Exploring more value prediction strategies
  – Example: Stride value prediction
    • Learning loop increments (e.g., [1,2,3], [2,4,6])
    • Similar to learning ["...page=1", "...page=2"] for URLs

• Predictor compression
  – Probabilistic classifiers

• Speculative execution of other types of agents
  – Example: Robot soccer agents