Parallelisieren einer Query Engine

Thomas Blum

Gliederung

1. Grundlagen
    1.1. Formen der Parallelisierung
    1.2. Operatorbaum
    1.3. NUMA
2. Konzepte
    2.1. Volcano
    2.2. Morsel-driven parallelism
3. Implementierung
4. Evaluation

Gliederung

1. Grundlagen

    1.1.Formen der Parallelisierung

    1.2.Operatorbaum

    1.3.NUMA

2. Konzepte

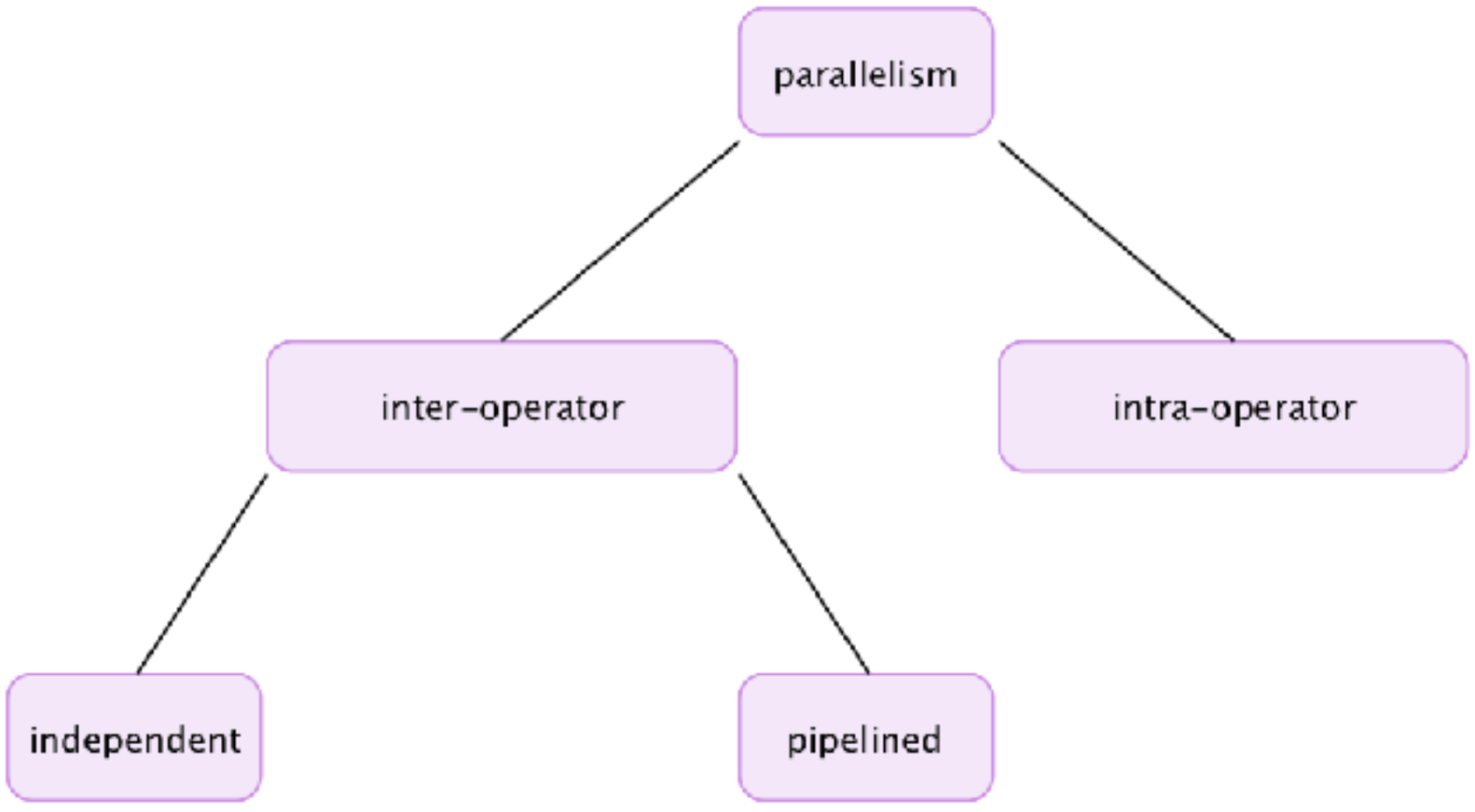
    2.1.Volcano

    2.2.Morsel-driven parallelism

3. Implementierung

4. Evaluation

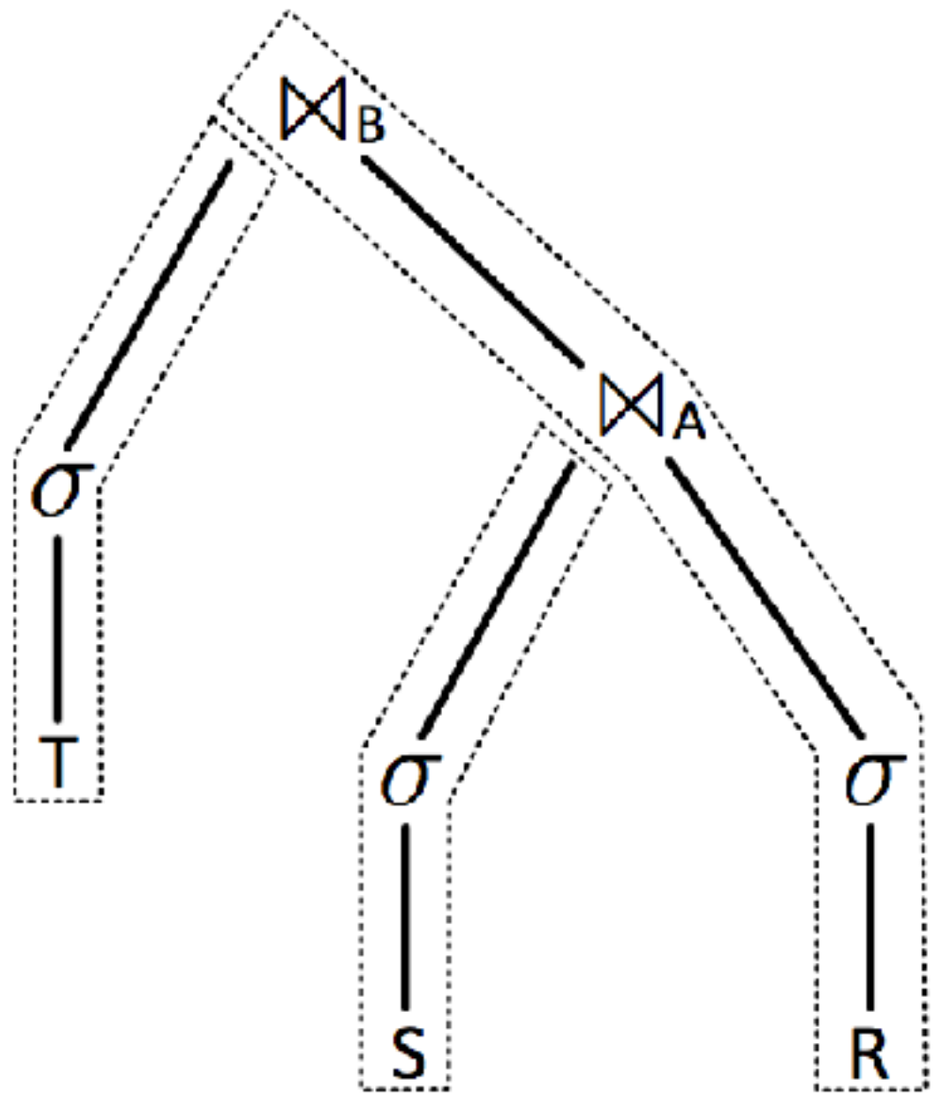
Formen der Parallelisierung



Gliederung

1. Grundlagen
    1.1. Formen der Parallelisierung
    1.2. Operatorbaum
    1.3. NUMA
2. Konzepte
    2.1. Volcano
    2.2. Morsel-driven parallelism
3. Implementierung
4. Evaluation

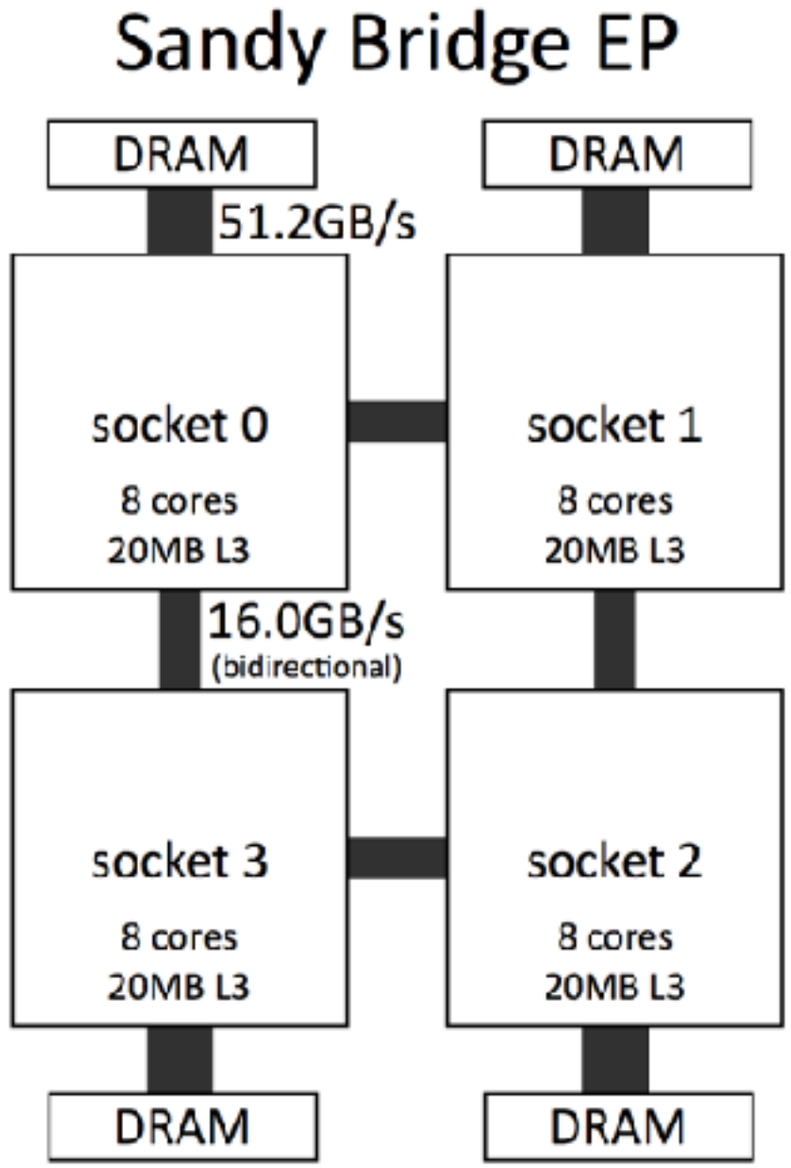
Operatorbaum



Gliederung

1. Grundlagen
    1.1. Formen der Parallelisierung
    1.2. Operatorbaum
    1.3. NUMA
2. Konzepte
    2.1. Volcano
    2.2. Morsel-driven parallelism
3. Implementierung
4. Evaluation

Non-uniform memory architecture



Gliederung

1. Grundlagen
    1.1. Formen der Parallelisierung
    1.2. Operatorbaum
    1.3. NUMA
2. Konzepte
    2.1. Volcano
    2.2. Morsel-driven parallelism
3. Implementierung
4. Evaluation

Volcano

- Abbilden von algebraischen Operatoren in die jeweilige Programmiersprache unter Implementierung des Iterator-Protokolls
- State records
    - Speichern Status-Informationen
    - Verknüpfen Operatoren
- Spezialfall: Exchange Operator zur Umsetzung der Parallelisierungsformen
    - Erzeugung neuer Threads
    - Verteilung der Tasks
    - Morgen der Ergebnisse

- Demand-driven dataflow
- Unabhängigkeit der Operatoren
- Iterator-Operationen:
  - open(): rekursives Aufrufen in Kindknoten, Initialisierung des state records, Erzeugung neuer Threads(Exchange Operator)
  - next(): Durchführung der Suche, Rückgabe eines NEXT_RECORD, Beenden der iteratives Aufrufen durch Rückgabe des end-of-stream indicator
  - close(): Beenden der Suche

Gliederung

1. Grundlagen
    1.1. Formen der Parallelisierung
    1.2. Operatorbaum
    1.3. NUMA
2. Konzepte
    2.1. Volcano
    2.2. Morsel-driven parallelism
3. Implementierung
4. Evaluation

Morsel-driven parallelism

- Erweiterung des Volcano Konzepts
- Berücksichtigung von NUMA
- Komponenten
  - QEPobject: Verwaltung des Ausführungsplans, Weiterleitung von ausführbaren Pipelines an Dispatcher
  - Morsel: Unterteilungseinheit des von einer pipeline zu verarbeitenden Speicherbereichs
  - PipelineJob: Spezifizierung der zu verarbeitenden morsel und der dazugehörigen pipeline
  - WorkerThread: Abarbeitung der PipelineJobs, persistent
  - Dispatcher: Liste mit zu verarbeitenden PipelineJobs

- Abbilden der Datenbanksuche in Operatorbaum
- Herausfiltern von Pipelines abgetrennt von Pipelinebreakern
- Übersetzen der Pipeline Funktionalität in Codefragment zur Vermeidung von Zwischenergebnissen
- Gewährleisten der NUMA Lokalität
- Elastizität durch work stealing
- Hauptziele:
  - Bewahren der NUMA-Lokalität
  - Volle Elastizität des Grades der Parallelisierung
  - Anpassung der Lastenverteilung

Gliederung

1. Grundlagen
    1.1. Formen der Parallelisierung
    1.2. Operatorbaum
    1.3. NUMA
2. Konzepte
    2.1. Volcano
    2.2. Morsel-driven parallelism
3. **Implementierung**
4. Evaluation

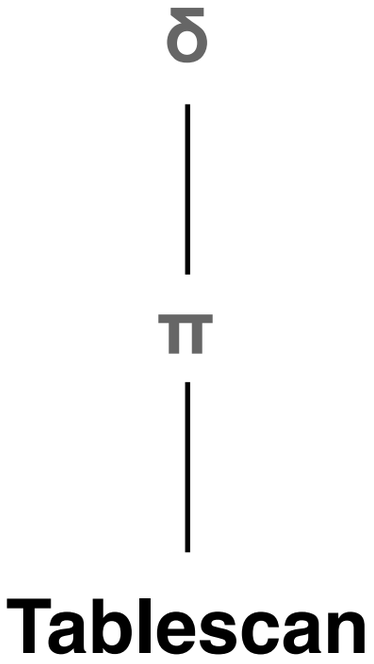
Gliederung

1. Grundlagen
    1.1. Formen der Parallelisierung
    1.2. Operatorbaum
    1.3. NUMA
2. Konzepte
    2.1. Volcano
    2.2. Morsel-driven parallelism
3. Implementierung
4. **Evaluation**

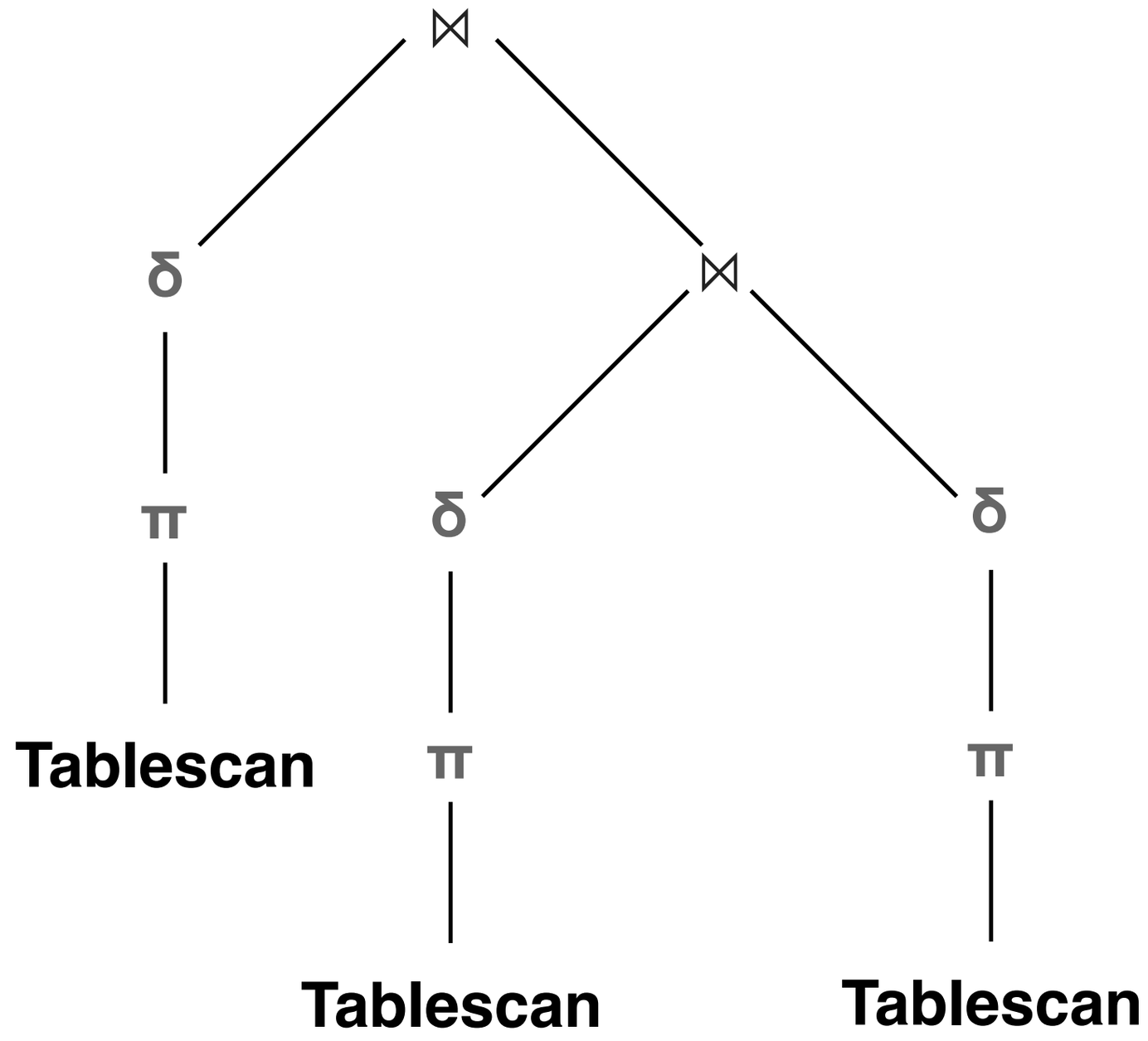
Evaluation

- Testbench:
  - 2,3 GHz Intel Core i7
  - 8 GB 1600 MHz DDR3

Evaluation

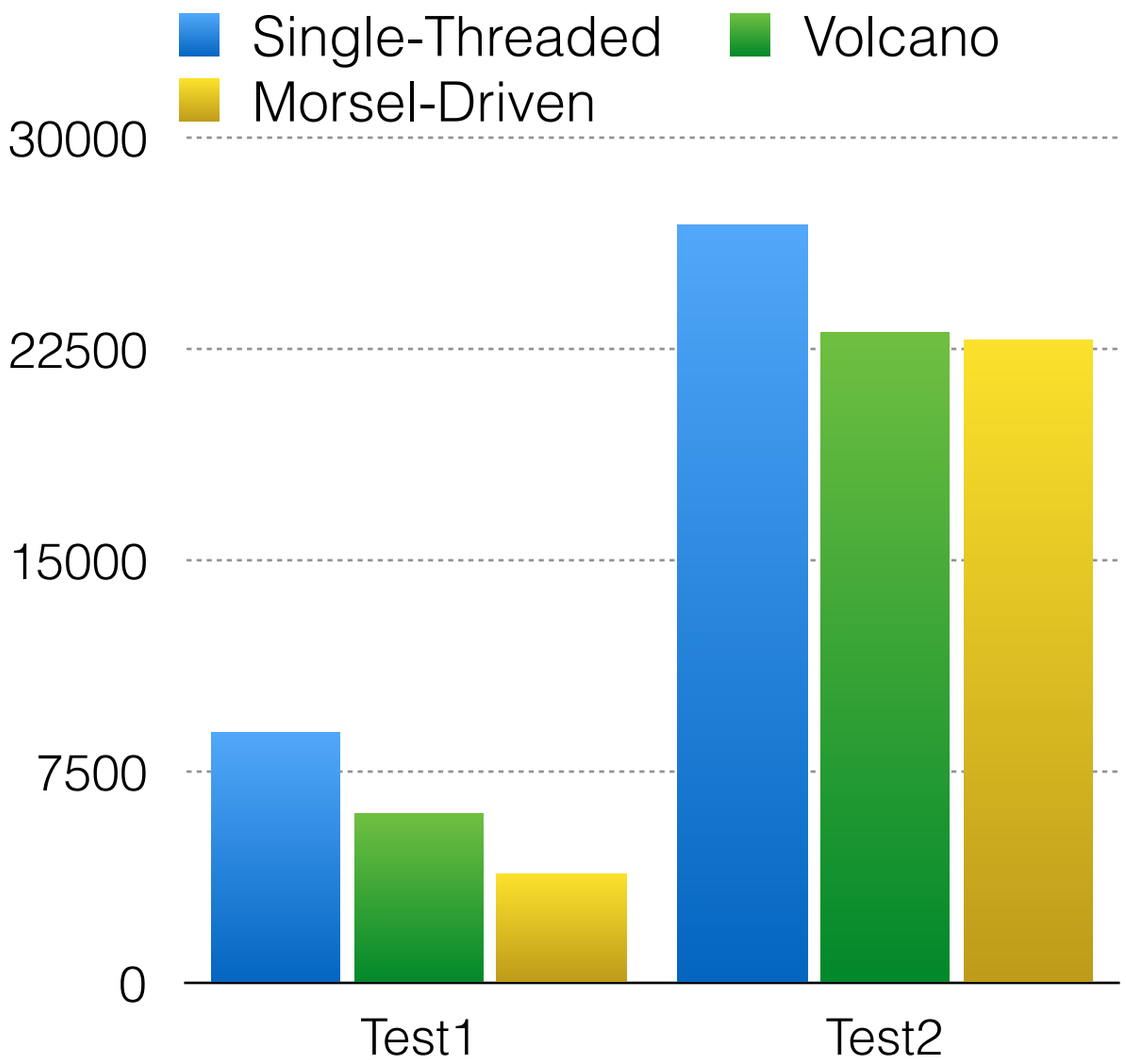


Test 1



Test 2

Evaluation


Single-Threaded
Volcano
Morsel-Driven
30000
22500
15000
7500
0
Test1
Test2

Q&A