High performance. Delivered.

Distributing Computationally Expensive Matching of Requirements to Capability Models

Reymonrod Vasquez, Kunal Verma, Alex Kass
Accenture Technology Labs

Introduction and Motivation

- Prior Paper:
  - U. Thayasivam, K. Verma, A. Kass, and R Vasquez. “Automatically Mapping Natural Language Requirements to Domain-Specific Process Models,” IAAI, 2011.

Overview

- Introduction and Motivation
- Overview of matching algorithm in ProcGap
- Performance on single machine
- Using caching to improve serial performance
- Overview of Apache Hadoop
- Implementing ProcGap on Hadoop
- Performance Analysis

Introduction and Motivation

- Fit/Gap analysis in large-scale ERP implementations
  - Identify whether capabilities of chosen software package fits the requirements
  - Identify any gaps

- Enterprises have developed reference process models
  - IBM's Web Industry Content Packs
  - Software AG's ARIS Reference Model
  - Accenture's Business Process Repository

Introduction and Motivation

- Tasks during solution development
  - Determine which portions of the reference models closely aligned to the client's requirements.
  - Identify capabilities in the reference model not defined in the client's requirements.
  - Identify requirements that are not defined in the reference model.

Introduction and Motivation

- Example of Capabilities and Requirements

Capability	Requirement
A. Create sales order	A4: Sales order shall be created by a user.
B. Create invoice	A1: The system shall allow the user to create a debit memo.
C. Create purchase order	A5: The system shall allow the user to enter discount code.
D. Maintain global price list	A2: The system shall the allow the user to set up a price table
E. Create delivery document	A3: The system shall allow the user to select transportation mode.

Introduction and Motivation

- Daunting task, currently performed manually
  - A typical requirements document in such projects contains more than 10,000 requirements
  - A typical process models contain 3,000 capabilities.

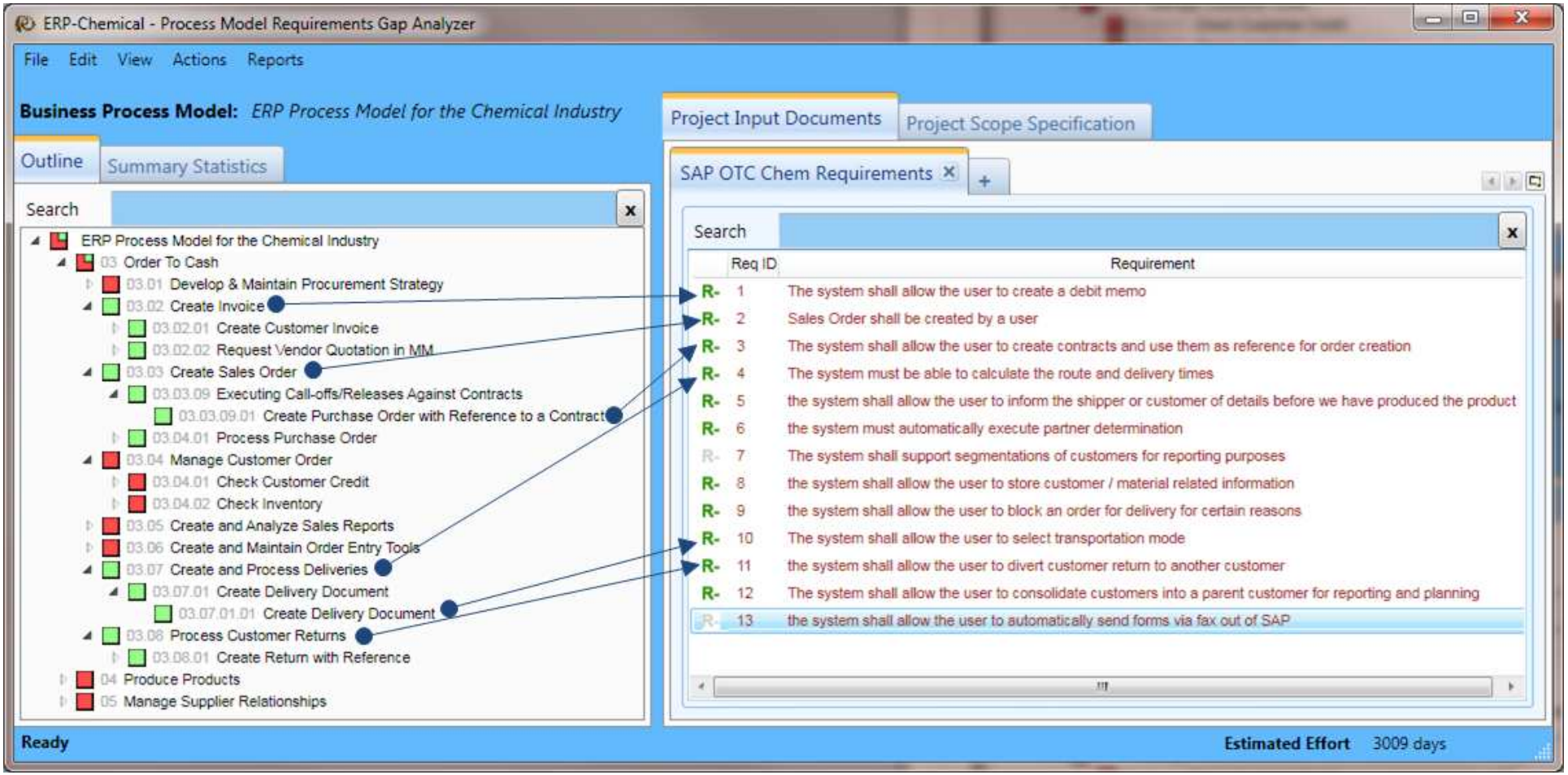
Introduction and Motivation

- Daunting task, currently performed manually
  - A typical requirements document in such projects contains more than 10,000 requirements
  - A typical process models contain 3,000 capabilities.

- Challenges:
  - Manual mapping is very tedious and time consuming.
  - Manual mapping is error prone and tends to produce inconsistent results.
  - Rely on text searches

Introduction and Motivation

- Process Model Requirements Gap Analyzer (**ProcGap**)
  - Automates mapping of requirements to capabilities



Introduction and Motivation

- Performance Issue:
  - It takes a long time to map large requirements documents to process models

Introduction and Motivation

- Performance Issue:
  - It takes a long time to map large requirements documents to process models

- Solution:
  - Caching mechanism
  - Use Apache Hadoop to distribute matching

Overview of matching algorithm in ProcGap

- Used Stanford parser to extract dependency trees from requirement and capability text.

Overview of matching algorithm in ProcGap

- Used Stanford parser to extract dependency trees from requirement and capability text.

- Use a set of heuristics to extract Verb-Object-Prepositional object (VOP) triples from the dependency trees.

Overview of matching algorithm in ProcGap

- Used Stanford parser to extract dependency trees from requirement and capability text.

- Use a set of heuristics to extract Verb-Object-Prepositional object (VOP) triples from the dependency trees.

- The extracted VOP's of requirement-capability pair are then compared using a combination of syntactic, semantic and information retrieval techniques.

Overview of matching algorithm in ProcGap

- Use of Cosine Similarity
  - For times when Stanford Parser cannot generate a phrase structure tree.
  - Create a word vector for the requirement and capability then use Cosine Similarity to calculate for a similarity score between the two word vectors.

Overview of matching algorithm in ProcGap

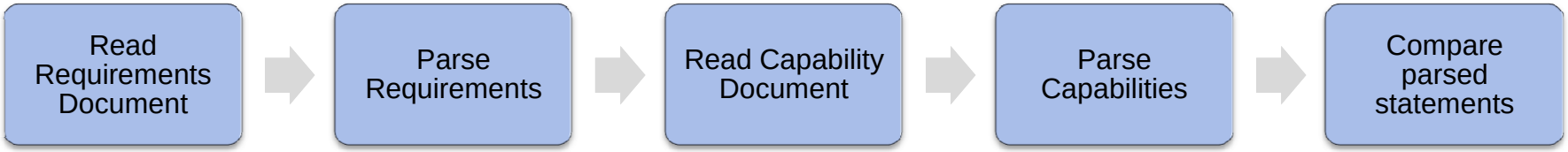
- Matching Examples

TABLE I. USING VERB-OBJECT-PREPOSITION IN MATCHING

	Capability	Mapped Requirement	Match Details	Match Types
1	Create Sales Order	System shall allow the user to create sales order	Verb: (create, create) via string match Object: (sales order, sales order) via string match	Syntactic VOP
2	Design and Develop Products	System shall allow user to inform customer of details before he produce the product	Verb: (develop, produce) Object: (products, product) via string match	Semantic VOP (using WordNet)
3	Enter invoice details	System shall allow the user to enter rebate rate in the forms	Verb: (show, enter) via record Object: (rebate, invoice) via sem_partof	Semantic VOP (using Semantic Graph)
4	Process Refunds	Support the process to refund revenue using workflow	{process, revenue, refund, workflow}; {process, refund}	Cosine Similarity

Performance on single machine

- ProcGap’s mapping tasks



Performance on single machine

- Matching performance
  - Dual core 3.0GHz CPU, 3.4GB RAM, Ubuntu Linux

Number of Requirements	Number of Capabilities	Matching time (seconds)
100	100	71
100	500	128
100	1000	160
100	2000	527
100	3000	3587

Using caching to improve serial performance

- A review of a large requirements document
  - A lot of the requirement statements share the same words.
    - Groups of sentences tend to focus on details of how a certain feature should work or behave.
    - For example a number of requirement statements may describe every possible path a user could take when interacting with a purchase order system. These statements will likely have a lot of words in common.

- A review of a process model hierarchy
  - A lot of its elements share the same words since they deal with aspects of the same domain.

Using caching to improve serial performance

- Introduced a Caching mechanism:
  - Store pair of words that reached the WordNet and Semantic comparisons, along with its result.
  - The cache is queried before proceeding with comparisons, avoiding the expensive comparisons
  - keep the cache small by not caching results of less-expensive string matching

Using caching to improve serial performance

- Sample of cache entries

Element 1	Element 2	Result
develop	Produce	WordNet match
show	Enter	WordNet match
rebate	Invoice	Semantic match

Using caching to improve serial performance

- Performance with caching

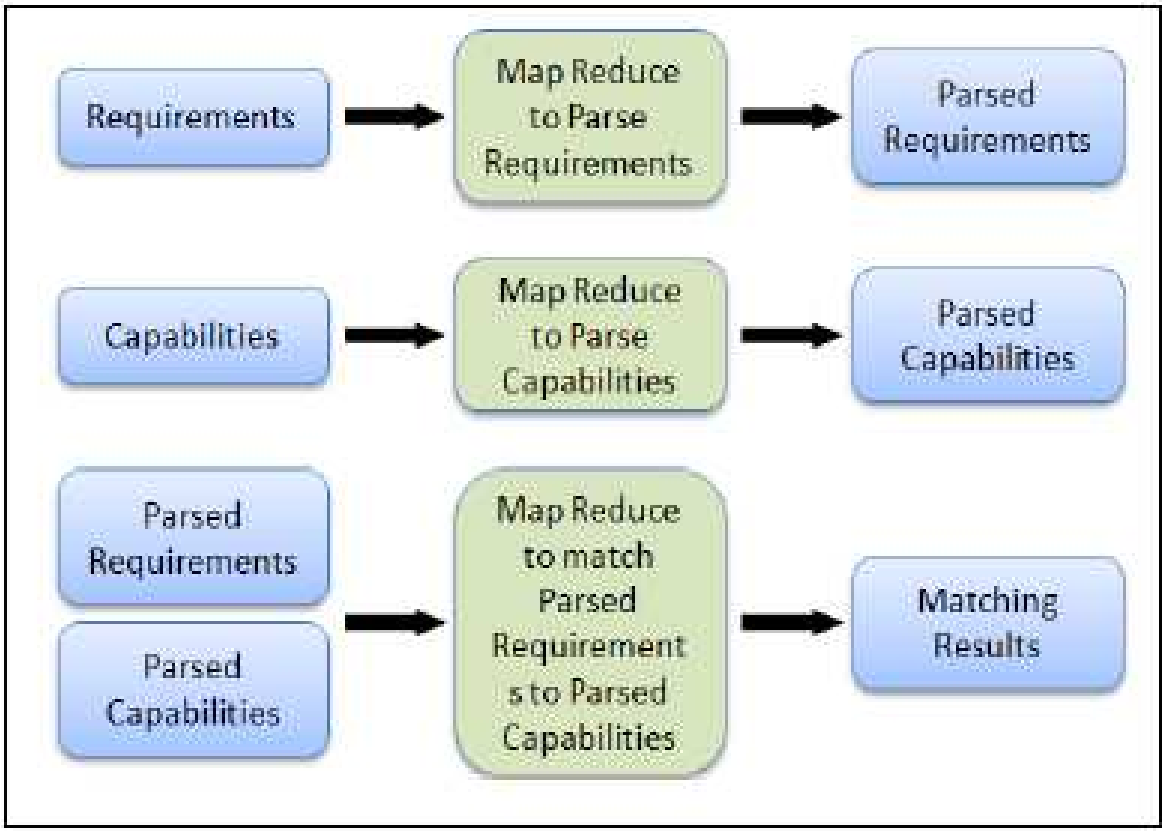
Number of Requirements	Number of Capabilities	Matching time (seconds)
100	100	60
100	500	87
100	1000	100
100	2000	163
100	3000	237
500	3000	403
1000	3000	941
2000	3000	10163
3000	3000	Out of memory

Overview of Apache Hadoop

- Hadoop is an open-source implementation of MapReduce, a programming model for processing very large data sets across many nodes.
- Data is distributed across a cluster of machines
- When executing code on the data, the code is transmitted to where the data resides

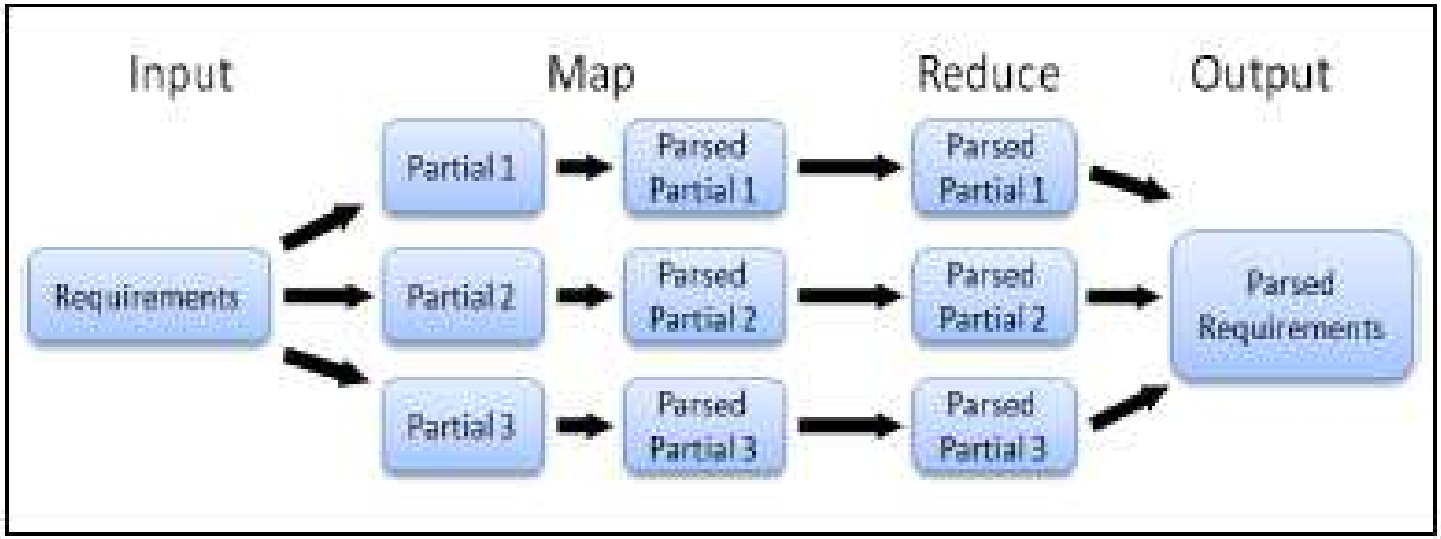
Implementing ProcGap on Hadoop

- ProcGap has 3 Map-Reduce executions



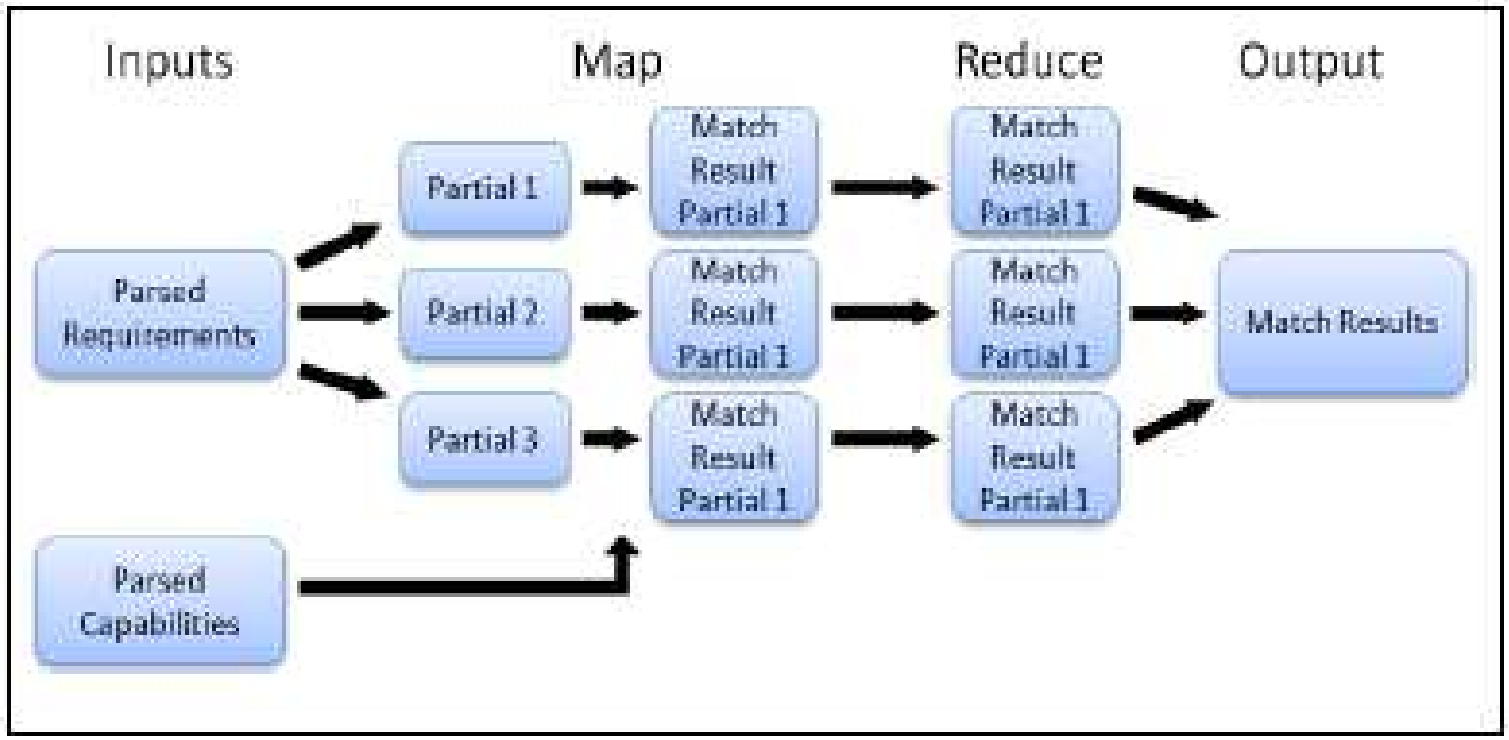
Implementing ProcGap on Hadoop

- Details of parsing the requirements using Map-Reduce



Implementing ProcGap on Hadoop

- Matching parsed Requirements to parsed Capabilities



Performance Analysis

- Runtime comparison

Number of Requirements	Number of Capabilities	Single Machine with caching Matching time (seconds)	4-node cluster With caching Matching time (seconds)
100	100	60	147
100	500	87	193
100	1000	100	212
100	2000	163	307
100	3000	237	251
500	3000	403	357
1000	3000	941	489
2000	3000	10163	896
3000	3000	Out of memory	614
5000	3000	-	903
10000	3000	-	2024

Thank You.

Questions?