Business Process Workflow Monitoring Using Distributed CBR with GPU Computing

Ioannis Agorgianitis[1], Stelios Kapetanakis[1], Miltos Petridis[2], Andrew Fish[1]

[1] Department of Computing, University of Brighton, UK, [2] Department of Computer Science Middlesex University, UK
[1]{i.agorgianitis, s.kapetanakis, a.fish}@brighton.ac.uk, [2]m.petridis@mdx.ac.uk

Abstract

Workflow monitoring and diagnosis can be a complex process involving sophisticated computational intensive operations. The ever-growing data generation and its utilisation have increased the complexity of workflow domains leading to an increased interest in distributed approaches for efficient workflow monitoring. Existing work has proposed a CBR enhancement to tackle deficiencies in areas where data volumes increase significantly. In such areas, the notion of a "data volume" component was proposed in an enhanced CBR architecture. This work proceeds further by evaluating a proposed distributed CBR lifecycle based on GPU programming to abstract further and evaluate the hypothesis that: increased data volumes can be tackled efficiently using distributed case bases and processing on demand. Our proposed approach is evaluated against previous work and it shows promising speedup gains. This paper signposts future research areas in distributed CBR paradigms.

Introduction

The complexity of enterprise applications along with their increased data generation and exploitation has led to the emergence of a "Big Data" era. Such systems encapsulate numerous interconnected business processes with sophisticated mechanisms for capturing, monitoring and managing continuous data streams and processes. However, in their vast majority they require human intervention to provide corrective actions in volatile points of the production lifecycle (Kapetanakis et al., 2010a).

Several standards have been developed with the aim of providing uniform mechanisms for handling business processes, such as BPMN (OMG, 2016). Existing work has shown a number of successful endeavours focusing on diagnosis and management of business workflows by using CBR (Aadmodt and Plaza, 1994) primarily as their core intelligence mechanism. (Kapetanakis et al., 2010a, 2010b, 2014) introduced an abstract architectural framework for CBR-based monitoring of business workflows, whilst representation and index-based retrieval of agile workflows have been introduced by Minor et al (2007). Towards workflow monitoring, Dijkman et al. (2009) developed a process-ranking model against a pool of process models.

Previous work (Agorgianitis et al., 2016) has introduced a new categorization of distributed CBR systems along with a distributed CBR lifecycle. Its results seemed promising and have presented increased performance gains. This work proceeds by abstracting and evaluating the proposed categorisation and CBR distribution with a substantially different distribution paradigm, that of GPU programming.

The paper continues by presenting the current trends in Big Data and ingestion of them by CBR along with a recap of our previous work which we use as a base-line metric. A brief illustration of the investigated domain is presented upon which the experiments are conducted. Our proposed distributed CBR approach (with GPU computing) is then presented along with its evaluation part, presenting the results of the experiments in comparison with a serial execution and horizontal distribution approaches. Finally, we conclude with a summary of potentials for distributed CBR in workflow management and monitoring.

Background Work

The ever-growing generation and capture of data in modern Information Systems, irrespective of application domain, seems a catalyst for great advancements in highly distributed systems. Berkley Open Infrastructure for Network Computing (BOINC) (Anderson, 2004) exploits idle time from heterogeneous web-enabled devices. Commodity-based distribution technologies like Apache Hadoop and Spark (Zaharia et al., 2010) perform massive distribution of processing across thousands of commodity computational nodes utilising both in-memory and on-disk processing schemes. Organizations and enterprises are migrating to such technologies to tackle an ongoing increased data volume (Netflix, 2014). Similarly, in scientific big data, a number of techniques are followed to increase effi-

ciency, scalability and migration of the various research domains to the big data world. Batselem et al. (2015) propose a reasoning approach for Resource Description Framework Schema (RDFS) that employs optimized methods based on Apache Spark. Jalali and Leake (2015) present a case study harnessing big data methods, specifically MapReduce and locality sensitive hashing (LSH), to make ensembles of adaptation for regression (EAR) approach feasible for large case bases without compression.

In previous work (Agorgianitis et al., 2016), we proposed a new categorization of distributed CBR systems (horizontal distribution), in which the data volume component was a key prerequisite in the integration of distribution in CBR systems specialized in business process workflow monitoring and management. A number of predicates were established upon which the following categorisations could take place: ground-up distribution; distribution of case base and processing on demand; and agent competency to handle large data volumes. The results of the experiments indicated that considerable performance gains can be achieved in specific areas of the CBR cycle in which increased data volumes can generate deficiencies. In this respect, the initial stages of the CBR cycle (case-loading, case-representation and similarity computations) could be largely distributed to enhance the performance of CBR systems via (GPU programming) whereas keeping the proposed distributed CBR lifecycle paradigm.

The CBR Domain

The utilised CBR domain comes from the area of ordering and distribution of goods to retail points involving workflow experts, data and business rules. The investigated business process comprises various phases, such as new order generation, order preparation through various departments and finally the dispatching and delivery of goods (see Figure 1) in a strictly timely manner. The required domain knowledge was acquired through past working experience within the domain throughout all departments of the workflow lifecycle.

Scaling CBR with GPU programming

Horizontal distribution handles increased workloads by adding and interconnecting hardware and software resources operating as "one" physical entity (interconnected servers). However, vertical distribution could also handle scaling, by utilising better each and every resource node by increasing the capacity of software and hardware resources in order to make it faster.

Aiming to provide an enhanced degree of abstraction, this work advances the state of the art by adapting a Vertical scaling approach. The evaluation is extended to cover the two major families of distribution, vertical and horizontal, thereby fully enhancing the volume-CBR lifecycle.

Vertical Scaling with GPU and CPU programming

The vertical distribution approach was conducted using a mixed distribution workflow. The raw data distribution and case loading are implemented using C# and Parallel LINQ (PLINQ) (Microsoft, 2016).

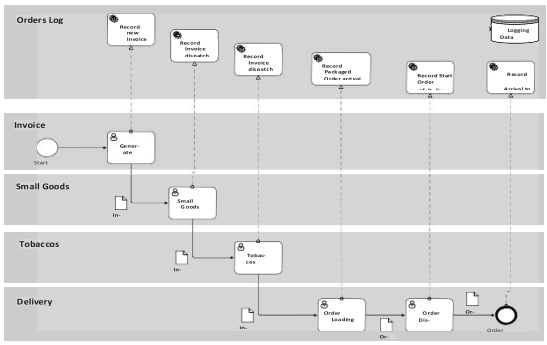

Figure 1. Business process definition (BPMN) of the investigated business process

The similarity computations take place by exploiting GPU parallel programming with NVIDIA CUDA kernel (NVIDIA, 2016) and the rest of the CBR cycle tasks operate using C# serial execution. The case bases developed in the horizontal scaling and serial execution experiments, text based for distributed versions and relational database for the serial execution, are both utilised in the vertical distribution. The Isomorphic Graph Similarity Links Algorithm

The business workflow of the current application domain is composed of a finite number of actions along with their corresponding intervals. Groups of actions and intervals are included in every valid instance to have a "complete" workflow. Action order is of importance and is fixed since the production phase is defined under specific schematics. As an example, "a dispatch order cannot take place before the generation of an invoice for this specific order".

An ordering process BPMN could be classified as isomorphic (Ruohonen, 2008) due to the fixed number of actions in conjunction with the constant way that the actions relate to each other. The lack of loop occurrences in the graph representation of the workflow means the graph is acyclic.

The Isomorphic Graph Similarity Links (IGSL) algorithm is a basic algorithm developed aiming to measure similarities of isomorphic and acyclic graphs specifically for our domain area. Given two acyclic and isomorphic Graphs G and G′, the similarity Sim(G, G′) between the two is calculated by:

$$Sim(G,G') = \sum_{i=1}^{i=count(E)} \sigma(E_i, E_i') / count(E)$$

where count(E) is the number of edges in G graph and $\sigma(E_i, E_i')$, with $0 \leq \sigma(E_i, E_i') \leq 1$, is the similarity measure between 2 individual edges E_i and E_i' from graphs G and G′ correspondingly.

The idea behind the development of the proposed algorithm is that given the fact that all workflow instances have the same number of nodes and the nodes are connected in the same way, the similarity between two given graphs could be calculated by measuring the distances of the corresponding links between the given graphs.

The GPU parallel programming model introduces the utilisation of GPUs in conjunction, depending on the number of cases up to 10 million. Having completed the similarity computations and their storage, along with the cases indices, the similarities are sorted and the K most similar indices are returned [Algorithm 1].

The K most similar indices are then used to retrieve the cases in question from a relational database. The retrieval is fast enough since the cases are indexed within the database. Finally, the rest of the CBR cycle (classification, adaptation and persistence) occurs in a sequential manner.

Algorithm 1. GPU - Similarity Computations

```
1: function ComputeKSimilarGPUHost
2:     Init Host vars and load data from text media (PLINQ)
3:     › graphsData, newCase, similarities
4:     Allocate the memory on the GPU and copy data
5:     › gpu.Allocate(graphsData, newCase, similarities)
6:     › gpu.CopyToDevice(allData)
7:     Launch 100.000 blocks of 1000 threads each
8:     › gpu.Launch(100000,1000).KernelSims(allData)
9:     Copy computed similarities from device
10:    › gpu.CopyFromDevice(similarities)
11:    return K most similar indices
12: function KernelSims
13: Compute thread Id
14:  › tId = threadId.x + blockId.x * blockDimension.x
15:   do while thread Id < Total Number of Cases
16:        Get graph data for each thread based on thread Id
17:        Compute IGSL similarity for current thread Id
18:        › sim[tId] = ISGL(currentGraphData, newCase)
19:  Update Thread Id so as to ensure exit form loop
20:  › tId = blockDim.x * thread.gridDim.x
21: function ISGL
22:   › similarity = 0
23:   for each edge1 in newGraph do
24:        for each edge2 in threadGraph do
25:             ›similarity += calculate edge1, edge2 distance
26:   return similarity
```

Evaluation

For the evaluation part, we assessed the proposed distributed CBR lifecycle and its level of abstraction by introducing distribution in CBR. We used heterogeneous distribution schematics, technologies and workflows in the application domain area of business workflow monitoring and diagnosis. Several experiments were conducted, integrating distribution in the initial stages of the CBR having the following hypotheses:

Hypothesis 1 *Increased data volumes can be tackled efficiently using distribution of case base and processing on demand, irrespective of underlying distribution frameworks and schematics.*

Hypothesis 2 *Increased data volumes and processing can be handled efficiently by a single agent entity*

The experimental runs were based on 7 real instance-based generated datasets. Each dataset contained log entries with information relating to actions (workflow-case) that occurred within the production phase. Each workflow included various actions from the log entries ranging from "invoice generation" to "delivery order". Fuzziness was introduced throughout our experimental data using 4 main categories of delays (no delay, minor, moderate and severe) along with application of domain knowledge delay rules and random fluctuations.

The classification of the cases was developed using a threshold classification scheme (50%). The actual evaluation of the distributed CBR lifecycle was conducted by developing a basic implementation of a k-NN classification algorithm in order to classify a new case fed into the system. The new case classification was delivered by a weighted voting of the k most similar cases.

The baseline metrics utilised in the current evaluation come from our previous work, including an optimised serial execution CBR cycle and a horizontal distribution approach using Apache Spark as the underlying distribution framework. The experimental runs involved a large variety of case numbers ranging from: 20 to 10 million. All experimental runs were conducted on the same machine (8 cores, 16 GB RAM, and a conventional NVIDIA display adapter with 48 CUDA cores).

Results

The results of the experiments are presented in Table 1 in contrast with our previous versions (serial and horizontal scaling executions) (Agorgianitis et al., 2016). The vertical distribution outer- performs the optimised horizontal scaling in each set of experiments.

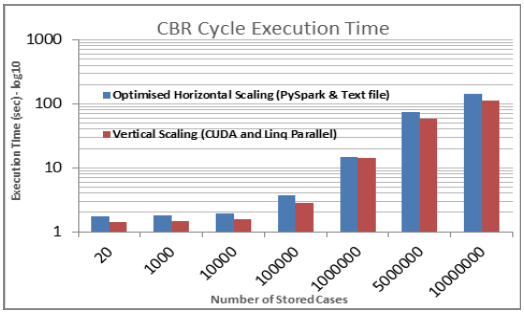

Figure 2. Vertical Scaling, PLINQ and CUDA Time Results

The break-even point, after which someone can notice performance gains, is between 10 and 100 thousand cases. Small numbers of cases had better performance via serial execution, due to distribution overheads of parallel approaches (see Figure 2).

A vertical distribution approach (GPU) involves more sophisticated execution workflows in comparison to the equivalent horizontal approach since it exploits two distributed technologies (PLINQ and CUDA parallel programming). This means that the produced overheads (due to parallelism) are greater, something which lead to performance downgrades. Furthermore, the utilisation of additional APIs in order to integrate different runtimes (CUDAfy) is another factor which introduces additional overheads to any vertical parallel execution.

Stored Cases	**Serial Execution** (sec)	**Horizontal Scaling** (sec)	**GPU Scaling** (sec)
20	0.279635668	1.725372791	1.4142951
10^3	0.370043755	1.789544344	1.4525411
10^4	0.782604933	1.943482876	1.5664515
10^5	6.322293282	3.685497522	2.8503591
10^6	58.5914855	14.62897038	14.4442463
$5*10^6$	1847.181649	73.43203068	57.708023
10^7	2370.668307	140.6282022	111.504969

Table 1. *Vertical Scaling, PLINQ and CUDA Approach Execution Time Results*

From our experiments it seemed like a vertical approach outperforms any horizontal distribution attempts. A maximum speedup of 32 is achieved for 5 million cases, which is a considerable improvement in performance. However, speedup appears to be downgraded for experiments with more than 5 million stored cases.

This observation may not seem valid at first glance since the similarity computations are massively parallelized (GPU fires up to 10 million concurrent threads in one go). The speedup decrease implies two things.

First, the similarity algorithm utilised in the experiments is a conventional one with a small degree of complexity. As a result, increased performance gains are prohibited due to the simplicity of the algorithm. Secondly, the distribution and loading of data are still bound to CPU processing which is limited by the number of the available cores. So, the bottleneck in terms of similarity computations is indeed resolved by using GPU programming but the one introduced by the data distribution and loading is still bound to the number of available cores.

Conclusions

This paper presents a novel approach in CBR workflow monitoring using distributed GPU programming. Superlinear speedup is observed for a high number of cases, indicating that CBR systems are bound to increase processing and I/O. Distribution is proven to provide performance gains in CBR systems tackling increased workloads and very large datasets. Further research will focus on formulating a generic architecture capable of incorporating distribution in CBR, irrespective of technical features, distribution schematics and application domains. Similarly our focus will be on data optimisation and process distributed pipelines and dynamic partitioning algorithms for large datasets.

References

Aamodt, A. and Plaza. E. 1994. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *Artificial Intelligence Communications* 7(1): 39-52

Agorgianitis I., Petridis M., Kapetanakis S., Fish A., 2016. Evaluating Distributed Methods for CBR Systems for Monitoring Business Process Workflows. In Proceedings of Workshop on Reasoning about time in CBR at 24th ICCBR 2016, RATIC, 2016, Atlanta, Georgia, USA: 122-131.

Anderson, D.P. 2004. BOINC: A System for Public-Resource Computing and Storage. In Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing

Batselem J., Young-Tack P.2015. Distributed Scalable RDFS Reasoning", BigComp 2015, pp. 31-34

Dijkman R.M., Dumas, M., Garcia-Banuelos, L. 2009. Graph matching algorithms for business process model similarity search. In Proceedings of the 7th International BPM. LNCS, 5701: 48–63

Jalali V., Leake D. (2015). CBR Meets Big Data: A Case Study of Large-Scale Adaptation Rule Generation", Proceedings of 23rd ICCBR 2015, Frankfurt, Germany, pp 181-196

Kapetanakis, S., Petridis, M., Knight, B., Ma, J., Bacon, L. 2010a. A Case Based Reasoning Approach for the Monitoring of Business Workflows. In Proceeding of the 18th ICCBR 2010, Alessandria, Italy, LNAI (2010)

Kapetanakis S., Petridis M., Ma J., Bacon L. 2010b. Providing explanations for the intelligent monitoring of business workflows using case-based reasoning. In Proceedings of the Fifth International Workshop on Explanation-aware Computing, ExaCt 2010, Lisbon, Portugal

Kapetanakis S., Petridis M., 2014. Evaluating a Case-Based Reasoning Architecture for the Intelligent Monitoring of Business Workflows. Successful Case-based Reasoning Applications-2, Studies in Computational Intelligence 494: 43-54 Springer-Verlag

Microsoft (2016) Parallel LINQ, https://msdn.microsoft.com/ Accessed November 16

Minor, M., Tartakovski, A. and Bergmann, R. 2007. Representation and structure-based similarity assessment for Agile workflows, In Proceedings of the 7th ICCBR 2007, Belfast. LNAI, 4626: 224–238. Springer-Verlag.

Netflix. 2014. Distributed Neural Networks with GPUs in the AWS Cloud. http://techblog.netflix.com/2014/02/ distributed-neural-networks-with-gpus.html,, Netflix Tech Blog, Accessed November 2016

NVIDIA. 2016. Parallel Programming and Computing Platform, CUDA, NVIDIA, http://www.nvidia.com/ Accessed November 2016

Object Management Group 2016. BPMN Version 2.0: OMG Specifications. From http://www.omg.org/ Accessed November 2016

Ruohonen, K. 2008. Graph Theory. Tampere University of Technology, Chapter 1, Section 5

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., Stoica, I. 2010. Spark: Cluster Computing with Working Sets. In Proceedings of the 2nd USENIX conference on Hot topics in cloud computing, pp.10-10