International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland

Performance-aware scheduling of streaming applications using genetic algorithm

Pavel Smirnov, Mikhail Melnik and Denis Nasonov

ITMO University, Saint-Petersburg, Russia

{smirnp, mihail.melnik.ifmo, denis.nasonov}@gmail.com

Abstract

The main objective of Decision Support Systems is detection of critical states and response on them in time. Such systems can be based on constant monitoring of continuously incoming data. Stream processing is carried out on the basis of computing infrastructure and specialized frameworks such as Apache Storm, Flink, Spark Streaming. However, to provide the necessary system performance at high load incoming data, additional data processing mechanisms are required. In particular, the efficient scheduling of streaming applications plays an important role in the data stream processing. Therefore, this paper is devoted to investigation of genetic algorithm to improve the performance of data stream processing system. The proposed genetic algorithm is developed and integrated into Apache Storm platform, and its efficiency is compared with heuristic algorithm for scheduling of Storm streaming applications.

© 2017 The Authors. Published by Elsevier B.V.
Peer-review under responsibility of the scientific committee of the International Conference on Computational Science

Keywords: scheduling, data streaming, apache storm, genetic algorithm

1 Introduction

Decision Support Systems (DSS) are becoming more and more essential part of the modern world. They find a place in our everyday life, as well as in Early Warning Systems (EWS), which are designed for monitoring, prediction and reaction on upcoming hazards and disasters, for example: for prevention of floods [1], in clinical practice [2] or for evacuation from damaged ships [3]. Integrated in DSS models use data to fit parameters and provide accurate predictions. The computation of such models must be completed in time to get the possibility to react on possible critical situations. Therefore, it is necessary for DSS to ensure the required computational environment which consist of heterogeneous computing resources, required software packages and middleware to manage environment. Deployed management system must not only organize the computational process, but also should optimize it. Effective scheduling mechanisms may lead to improved performance or reduced energy costs. The choice of optimization criteria depends on application. The data for analyzing may come with a certain periodicity which must be processed for specified period of time,
that is understood as the batch processing. On the other hand, data can flow in a continuous stream of incoming data, that should be processed online. In this cases, we can talk about data streaming and corresponding streaming applications.

Today the abilities of Big Data technologies allow to process gigabytes of data, receiving from devices in real-time. The stack of data-streaming technologies like Apache Storm, Flink, Heron and others, emerged during last 5 years, demonstrates actual needs in high-performance and scalable real-time data processing. Traditional Big Data tools like Hadoop, Spark, Hive, Pig and others are designed as batch processing engines and oriented to process a fixed amount of data, accumulated and prepared in advance. Stream processing model serves to continuous data flows instead. It means that large amounts of data are arriving dynamically and are impossible to be evaluated in advance. The difference between the two models leads to a difference in scheduling and resource-allocation approaches.

Subject of the paper is devoted to scheduling algorithms for real-time data streaming systems. The contribution of the paper is performance-aware scheduler based on genetic algorithm (GA). The algorithm is implemented as a custom scheduler for Apache Storm and its' efficiency compared with other Storm scheduling algorithms and experimentally proved. The remainder of the paper is organized as follows. In the second section, we consider several works, which related to scheduling of streaming applications. The third section provides the background of Apache Storm framework and the definition of optimization problem. Proposed GA with its operators and models for evaluation is presented in section 4 with another stream scheduling algorithm R-Storm, which is used in our experimental study for comparison. Section 5 consist of experiments, which were conducted to investigate the performance of our proposed approach, and the conclusion is presented in section 6.

2 Related works

Scheduling approaches for real-time data processing basically differ from traditional batch (workflow) scheduling ones. During a literature overview we outlined number of points (see table 1), which define difference between the two kind of approaches.

<table>
<thead>
<tr>
<th>Schedule moment</th>
<th>Batch</th>
<th>Streaming (realtime)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheduling time (s)</td>
<td>Non critical (up to 30 seconds)</td>
<td>Critical (seconds, subseconds)</td>
</tr>
<tr>
<td>Scheduling policy</td>
<td>Assign computation to the nodes where the required data are stored</td>
<td>Assign most communicating tasks together on one node or rack</td>
</tr>
<tr>
<td>Input data</td>
<td>Static amount, annotated in advance (before start)</td>
<td>Continuous flow, variability of data, impossible to get in advance</td>
</tr>
<tr>
<td>Resources</td>
<td>Allocated statically</td>
<td>Allocating dynamically</td>
</tr>
</tbody>
</table>

Table 1: Difference between batch and streaming data processing

Default Storm’s scheduler implements a primitive round-robin strategy, which iterates tasks and assigns them to node’s slots equally. The strategy is far from optimal for heterogeneous cluster and different tasks’ workload. Several papers propose custom scheduling algorithms to improve the performance of streaming applications. The comparison of algorithms is presented in the table 1.

In [4] authors proposed static and dynamic scheduling algorithms called offline and online respectively. The key idea of both algorithms is to use communication patters among executors trying to place the most communicating executors to the same slot. The algorithm is effective for topologies, where computation latency is dominated by tuples transfer time and performance improvements
دریافت فوری متن کامل مقاله

امکان دانلود نسخه تمام متن مقالات انگلیسی
امکان دانلود نسخه ترجمه شده مقالات
پذیرش سفارش ترجمه تخصصی
امکان جستجو در آرشیو جامعی از صدها موضوع و هزاران مقاله
امکان دانلود رایگان ۲ صفحه اول هر مقاله
امکان پرداخت اینترنتی با کلیه کارت های عضو شتاب
دانلود فوری مقاله پس از پرداخت آنلاین
پشتیبانی کامل خرید با بهره مندی از سیستم هوشمند رهگیری سفارشات