DISS. ETH NO. 23759

Astronomy and Computing

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZURICH

(Dr. sc. ETH Zurich)

presented by

STEFAN CHRISTIAN MÜLLER

Master of Science ETH in Computer Science, ETH Zurich

born on 18.03.1982

citizen of Wädenswil, Zürich

accepted on the recommendation of

Prof. Dr. Gustavo Alonso, examiner
Prof. Dr. André Csillaghy, co-examiner
Prof. Dr. Timothy Roscoe, co-examiner

2016

Abstract

The data volume created in data heavy sciences such as astronomy is growing at an exponential rate. Drawing conclusions from data requires extensive processing, which requires high performance computing infrastructures.

In astronomy, the processing needs are diverse. Each instrument is novel and unique, producing new data in new data types. The required processing is as diverse as the research questions the data scientists are interested in. As we learn more about our universe, scientists move on to new questions, requiring new data and new ways of processing. As a result, a large portion of the processing and analysis code is short lived. In exploratory data science, code is written not with the purpose of creating a software product, but as a means to better understand the data. Such code typically changes for each execution run since new information is gained each time, with leads to immediate changes to the code. This development process is comparable to software prototyping, only that a small test dataset is rarely sufficient and large data volumes need to be processed for each iteration.

This calls for systems that can utilize high performance infrastructures without demanding significant development efforts to achieve parallelization. If the process has to be formulated in a constrictive pattern, such as MapReduce, or rewritten in a special language, it is often not efficient to employ such a system, as these efforts cannot be amortized over a long lifetime of the software, as it is possible in many business use-cases for big data. For example, most data warehouse systems assume that the data can be represented as OLAP cubes and that queries can be answered by filtering and aggregating. Since this is the case for many practical applications across many companies, vendors can invest significantly into the development of such systems. Astronomy data processing is lacking these economies of scale.

This dissertation presents Pydron, a system that takes the volatile nature of the code

into account. Instead of focusing entirely on minimizing the execution time, Pydron is designed to allow for a fast development cycles, which includes the time to set up the system and write code, while scaling to large high performance infrastructures, such as clouds. This is achieved by semi-automatically parallelizing sequential, single-threaded, Python code. This approach significantly reduces the economic barriers to utilize highly parallel infrastructures, as it avoids the cost of reimplementation in a different language and is flexible enough to handle arbitrary Python code, avoiding a constraining framework. The system is designed to be quickly deployable on a wide range of infrastructures to ensure that existing resources can be employed.

Python is currently the language of choice in astronomy. It has proven to be an effective and convenient tool to write astronomy data processing and analysis code. With Pydron the data scientists can keep using their language of choice, while profiting from the performance of modern infrastructures. With a small API consisting of only two decorators, Pydron is easy to use, even with existing code. Behind the scenes, Pydron handles parallelization, scheduling, and resource management automatically, allowing the data scientists to focus on their research.

Zusammenfassung

Die Datenvolumen welche in Wissenschaften mit einem Schwerpunkt auf Daten gesammelt werden, wächst exponentiell. Um Schlussfolgerungen aus den Daten zu ziehen, müssen diese umfangreich verarbeitet werden. Dazu sind Hochleistungsrechnerinfrastrukturen notwendig.

In der Astronomie sind die Bedürfnisse für die Datenverarbeitung vielseitig. Jedes Instrument ist neuartig sowie einzigartig und erstellt neue Daten in neuen Datenformen. Die benötigte Datenverarbeitung ist so vielseitig wie die Forschungsfragen der Datenwissenschaflter und Wissenschaftlerinnen vielseitig sind. Mit fortschreitendem Wissen über unser Universum kommen neue Fragen in den Fokus der Wissenschaft, welche wiederum neue Daten und neue Datenverarbeitung benötigen. Entsprechend sind grosse Teile vom Datenverarbeitungs- und Datenanalysecode kurzlebig. Programmcode für explorative Datenanalyse wird nicht mit dem Ziel geschrieben ein Softwareprodukt zu erstellen, sondern dient als Hilfsmittel um die Daten besser zu verstehen. Solcher Programmcode ändert sich typischerweise für jede Ausführung, da jedes Mal neue Informationen gewonnen werden, welche wiederum zu Codeänderungen führen. Dieser Entwicklungsprozess ist vergleichbar zu Prototypenentwicklung von Software, allerdings ist ein kleiner Testdatensatz selten ausreichend, und grosse Datenmengen müssen für jede Iteration verarbeitet werden.

Dies verlangt nach einem System das Hochleistungsinfrastukturen verwenden kann, ohne erheblichen Entwicklungsaufwand zu verlangen, um parallele Ausführung zu ermöglichen. Wenn die Datenverarbeitung in einem einengendem System wie MapReduce formuliert werden muss, oder gar in einer speziellen Sprache neu geschrieben werden muss, ist es oft nicht effizient so ein System einzusetzen, da dieser Aufwand nicht über eine lange Lebensdauer der Software amortisiert werden kann, wie es für kommerzielle Anwendungen im Big Data Bereich möglich wäre. Die meisten Data-Warehouse Systeme nehmen an, dass die Daten als OLAP-Würfel dargestellt werden können, und dass Filtern und

Aggregation ausreicht um die Anfragen zu beantworten. Da dies in vielen kommerziellen Anwendungsfällen in vielen Firmen der Fall ist, können die Hersteller solcher Systeme erhebliche Ressourcen in die Entwicklung investieren. Der astronomischen Datenverarbeitung fehlt dieser wirtschaftliche Grössenvorteil.

Diese Dissertation präsentiert Pydron, ein System welches die hohe Änderungsrate des Programmcodes miteinbezieht. Anstatt sich ausschliesslich auf die Reduzierung der Ausführzeit zu konzentrieren, ist Pydron für ausgelegt durch Skalierung auf Hochleistungsinfrastrukturen schnelle Entwicklungszyklen zu erreichen. Pydron zieht die Zeit welche benötigt wird um das System einzurichten und den Programmcode zu schreiben mit in Betracht. Dies wird durch semiautomatische Parallelisierung von sequentiellem, einzel-Thread Pythoncode erreicht. Diese Vorgehensweise reduziert die ökonomischen Hemmschwellen um hoch parallele Infrastrukturen zu verwenden erheblich, da es die Kosten einer Neuimplementation in einer anderen Sprache vermeidet, und genügend flexibel ist, um mit beliebigen Pythoncode umgehen zu können und somit ein einschränkendes Programmiergerüst vermeidet werden kann. Das System ist so ausgelegt, dass es schnell auf einer grossen Vielfalt von Infrastrukturen in Betrieb genommen werden kann. Dies stellt sicher, dass bestehende Ressourcen verwendet werden können.

Python ist im Moment die Sprache der Wahl in der Astronomie. Sie hat sich bewährt als effektives und praktisches Werkzeug um Datenverarbeitungs- und Datenanalysecode für Astronomie zu schreiben. Pydron erlaubt den Datenwissenschaftlern und Wissenschaftlerinnen sowohl ihre Wahlsprache zu verwenden als auch die Leistung moderner Infrastrukturen zu benützen. Mit einer kleinen API aus nur aus zwei Dekoratoren ist Pydron einfach zu verwenden, auch mit bestehendem Code. Hinter den Kulissen kümmert sich Pydron automatisch um die Parallelisierung, Disposition und Ressourcenmanagment, damit die Datenwissenschaftler und Wissenschaftlerinnen sich auf die Forschung konzentrieren können.