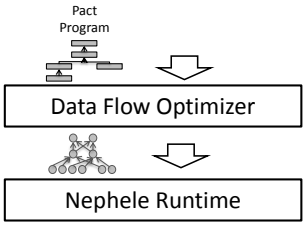

Figure 1: High-level architecture of Stratosphere

In our demonstration, we highlight the flexible and convenient support of PACTs to achieve this objective. We show how the Stratosphere system optimizes and executes the resulting programs in a massively parallel way and we emphasize the performance gains this provides. We exemplify the potential of this approach by offering an explorative tool for analyzing the dynamics of the role composition of an online community in a scale for which alternative solutions emerge as infeasible. As such, our demonstration will act as a showcase and practical guideline for other researchers that aim for huge-scale analytics of social media and online communities – and can achieve this by relying on the flexible support of PACTs offered by the Stratosphere system.

2. PACTS IN STRATOSPHERE

Stratosphere is an open-source[2] system that enables the massively parallel execution of data analytics tasks. This demo is based on the publicly available release version 0.2 and focuses on how to specify a complete use case as an analytics pipeline in Stratosphere. Other Stratosphere related demos focused on the efficient execution of incremental iterations [4] and the reordering of operators [6]. Both system capabilities are not part of the current open source version of Stratosphere and might become part of future releases. The application of Stratosphere's declarative query language Meteor to complex analytics tasks was demonstrated in [7].

The overall architecture of Stratosphere is depicted in Figure 1. The basic layer is provided by the parallel execution engine *Nephele*. It executes data flow programs modeled as directed acyclic graphs (DAGs) in a parallel and fault-tolerant way. Nephele keeps track of task scheduling and setting up the required communication channels. In addition, it dynamically allocates the compute resources during program execution.

A user of the Stratosphere system formulates algorithms in Stratosphere in one of multiple programming interfaces, which support different levels of abstraction. The optimizer chooses the physical execution strategies for the individual functions with the objective of minimizing the overall execution costs.

In order to leverage Stratosphere for truly large-scale community analytics, we propose that analysts should have to focus on only creating appropriate plans of *Parallelization Contracts (PACTs)*. PACTS are essentially a similar level of abstraction as the popular MapReduce paradigm. The user implements functions that the system evaluates on partitions of the data, where special second-order functions define how these partitions are actually formed. In addition to Map and Reduce, PACT offers binary second-order functions, such as Match, CoGroup, and Cross. These contracts can be freely assembled into directed acyclic graphs which allows for the creation of complete analytics pipelines including feature extraction, model training and post-processing that can be implemented as UDFs inside the second order functions.

[2]The Stratosphere system is open source under the Apache License, Version 2.0. and can be downloaded from `http://www.stratosphere.eu`

3. DEMO SCENARIO: ROLE ANALYSIS

For the actual demo scenario, we have selected role analysis as one of the key analytics that can provide valuable insights into the evolution and state of online communities and their members. We have implemented the behavior analysis approach proposed by [2] and refined by [8], which analyses the percentages of different roles that are assumed by community users over time. To do this, the behavior that users exhibited are measured by extracting numeric features from the data, each intended to capture one particular dimension of a user's behavior (e.g., focus dispersion, popularity, engagement, initiation or contribution). This feature extraction is a data-intensive task, since the features need to be computed for each particular individual over time from the raw data. To achieve a fine granular analysis over time, this is usually done on a daily or weekly basis for each sub-community (e.g., all different forums) separately. In each step of time, the activities and interactions of each individual in the recent past are considered for the feature computation, where both of the before mentioned works apply a sliding window of 6 months.

Based on the computed features the approach clusters users to deduce the role composition for each sub community at a given point in time. The role mining and inference process requires the iterative application of clustering methods over time to extract the behavior roles that emerge from the data. Additionally, once the roles are identified, an inference process is put in place to derive role labels for each particular user over time. Note that this behavior analysis approach is not static and assumes that the role of a user is contextual (i.e., it depends on her actions and interactions observed at particular time steps). This means, for example, that a user can adopt the role of "novice user" the first time she registers with a particular online community, and achieve the role of "expert" months after. The dynamics of the role mining and inference process, involving continuous clustering of the whole user base, constitute a major bottleneck of the approach when applied to large-scale datasets.

For example, this approach has been previously applied to a community dataset with 95,200 threads, 421,098 posts and 32,942 users. After introducing intensive manual code and database optimizations (e.g., multithreading, database load on demand, indices, etc.) the execution of the whole workflow on an Intel(R) Xeon(R) CPU machine with 2 processors at 2.27GHz and 24 GB of memory took 4.5 hours for the feature extraction and more than 16 hours for the role mining and inference process. These numbers illustrate that this approach is limited by the significant computational cost. A delay of hours of processing does not permit the extraction of rapid insights that community managers may need to maintain the health of their communities, neither is it feasible to support an explorative analysis for the involved researcher. Exploiting the massively parallel Stratosphere system for this task allows us to run this sort of analysis on data sets that were simply not feasible before. As such,

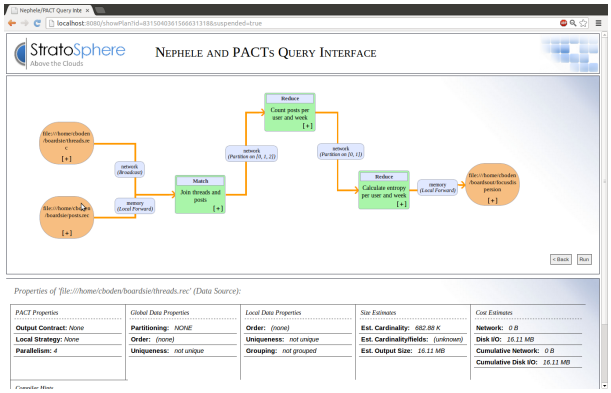

Figure 2: Screenshot of the optimized physical Execution Plan of the PACT Plan listed in Algorithm 1

pipeline part	processing time
preprocessing	1.3 minutes
feature extraction	10.3 minutes
clustering	27.5 minutes

Table 1: Average execution times on our research cluster consisting of 26 machines

we are able to now process the whole data from the most popular Irish discussion site boards.ie.[3] This encompasses 8.26 million posts in 680,000 threads created by 138,000 different users over 10 years of complete data, a significantly larger data set than in all our experiments before. Table 1 shows the achieved runtimes for pre-processing, feature extraction and clustering required for the role composition for each user and each week. These were computed for each forum separately and in addition over all forum borders taking all global user activity into account. This addition provides further valuable insights into the evolution and state of the boards.ie community as a whole.

It is important to note that this approach for community analysis resembles close similarities to a wide range of other analytical tasks. As such, it is an example tailored to showcase the general applicability and the gained benefit from processing such tasks in the Stratosphere system.

Algorithm 1: Simplified PACT example for extracting one feature

```
1 MatchContract joinThreadsPosts(threads,posts,postID)
2 ReduceContract countPosts(joinThreadsPosts, keys:
  {forumId,users,week})
3 ReduceContract calculateEntropy(countPosts, keys: {week,user})
4 output FileDataSink FileDataSink(calculateEntropy)
```

Algorithms 1 and 2 illustrate how convenient and easy it is to formulate the analytics pipelines consisting of PACTS. Algorithm 1 shows an exemplary sub plan consisting of several different parallelization contracts which extract the FocusDispersion feature as suggested in [8]. Each of these contracts contains user code to actually carry out a certain subtask. This is illustrated by Algorithm 2, which lists the UDF implemented inside the *calculateEntropy()* Reduce contract from line *3* in Algorithm 1. A plan is assembled by instantiating the contracts and chaining the inputs such that subsequent contracts consume the output of their predecessors. The contracts needed for preprocessing the data and clustering the users can be added to the PACT plan just as easily so that the complete workflow can be expressed as one PACT plan.

Algorithm 2: `calculateEntropy(record,collector)`

```
1  int sum = 0
2  List<Integer> grouped = Lists.newArrayList()
3  while records.hasNext() do
4  |   PactRecord record = records.next()
5  |   userId = record[0]
6  |   week = record[1]
7  |   int count = record[2]
8  |   sum += count
9  |   grouped.add(count)
10 double h = 0.0
11 forall the Integer group : grouped do
12 |   double ratio = ((double) group) / sum
13 |   h += ratio * Math.log(ratio)
14 entropy.setValue(-h)
15 outputRecord[0] = userId
16 outputRecord[1] = week
17 outputRecord[2] = entropy
18 collector.collect(outputRecord)
```

Figure 2 shows the FocusDispersion plan outlined in Algorithm 1 a PACT plan and illustrates the execution strategies chosen by the optimizer. Each vertex corresponds to a PACT, containing the user-defined function and a strategy to apply it to the data, according to the second-order function. Edges represent the result of one PACT being forwarded to the next. They are labeled with the data-shipping strategy they execute (e.g., in-memory pipe or network hash partitioning) and local operations they apply (such as sorting). In addition, the plan shows which data structure is used to store and update the solution set. A click on a vertex shows inferred properties of the data, such as partitioning and order properties, as well as basic cost estimates for disk and network usage.

4. THE DEMO ITSELF

The demonstration comprises three main parts:

1. Formulation of complex analytical tasks by means of understanding and writing PACT execution plans.
2. Running and monitoring the optimization and execution of the resulting data flows.
3. Exploration of the results gathered with the implemented large-scale role analysis.

We believe that this variety of aspects will offer at least one interesting facet for each conference attendee.

The objective of the first part, the formulation of PACT plans, is to illustrate the intuitive and convenient approach for expressing complex data-analysis workflows using PACTs. We will explain the different parts of the role analysis code and allow interested attendees to change parameters and even code. As such, they can learn how different data sets can be read and processed, how different features can be computed, and how other mining tasks can be expressed. This could even involve data provided by interested attendees – given that they are willed to share them and that an adaption of the import procedure is achievable in reasonable time (i.e, the data is structured similar to the boards.ie data). In limits naturally imposed by the demonstration setup, this can involve the execution of modified code to

[3]http://boards.ie

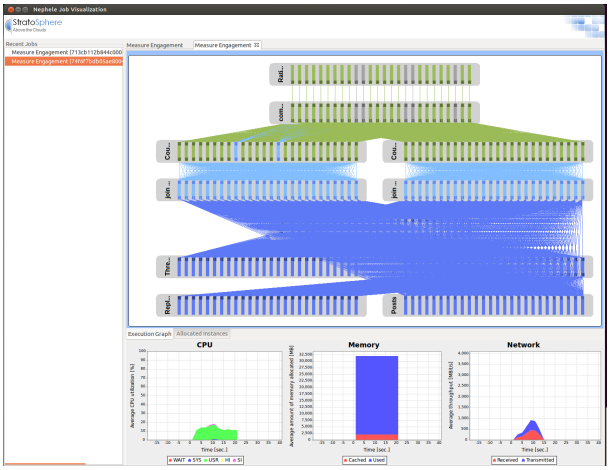

Figure 3: The Execution Monitor shows the data flow exectuted by the Nephele engine and also illustrates the degree of parallelism of each task.

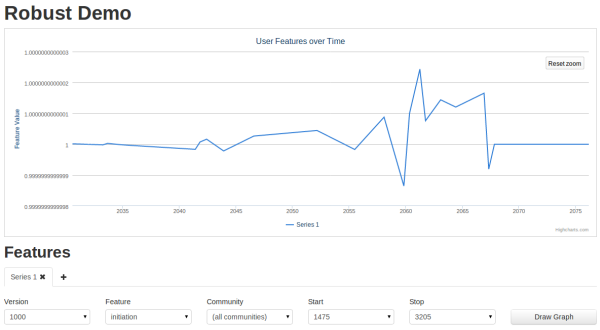

Figure 4: Explorative analysis tool

leverage insights gained from the second and third parts of the demonstration. This will mainly require smaller data sets so that jobs on the cluster can finish in reasonable time and code modifications that do not require extensive debugging.

The second step, running and monitoring the written PACT programs, aims at conveying internals of the Stratosphere system in a demonstrative way to the audience. This will illustrate how the PACT plans are optimized and transformed into data flows that are finally processed on our remotely available cluster. While running, we will make use of Stratoshpere's monitoring tools. An example illustration of a PACT program is shown in Figure 2, while Figure 3 shows an example screenshot of the execution monitor. The tool visualizes the parallel data flow with its vertices and edges. Vertices inside a group execute the same operations on different partitions of the data. The color of a vertex indicates its status, such as executing or finished. Further, the execution monitor provides several graphs that show multiple performance metrics and statistics about the job. This will illustrate how the complex tasks are distributed over nodes, which resources are involved at what point in time, etc. While it will not be feasible to complete the processing of the whole boards.ie data, we will be able to demonstrate the complete processing of smaller data sets – potentially with PACT programs modified or created in the first part of the demo. For illustrating the gained speed-up, the same tasks can be started and monitored on the local machine or a single node of the cluster.

Finally, the third part of the demonstration will highlight the benefits of processing an analytical task as complex as the implemented role analysis on a massively parallelized system. We will provide a visualization tool that allows to interactively explore the pre-computed results over the whole boards.ie data set over all 10 years. Attendees will be able to explore the dynamics of role compositions over the complete lifetime of the boards.ie site, with different parameter settings, over different time spans, different forums or the global boards.ie data, etc. Figure 4 shows a screenshot of this explorative interface, which supports the interactive visualization of different (normalized) features as well as role compositions over arbitrary time intervals and forums. An analysis like this was not feasible before due to the immense computational requirements it imposes. Thus, it is tailored to showcase the benefits that the proposed approach based on Stratosphere brings for the field of social-media and online-community analytics.

Acknowledgements

We thank Christoph Nagel and Stephan Pieper (now with `http://www.surpreso.com`) for their implementation support while at TU Berlin and the Stratosphere team. The research leading to this result was funded by the European Commission under FP7 Project No. 257859 - 'ROBUST', the German Research Foundation under grant FOR 1036, the German Federal Ministry of Education and Research (BMBF) under grant number 01ISI2033 - 'RADAR' and the European Institute of Innovation and Technology (EIT).

5. REFERENCES

[1] D. Battré, S. Ewen, F. Hueske, O. Kao, V. Markl, and D. Warneke. Nephele/PACTs: A Programming Model and Execution Framework for Web-Scale Analytical Processing. In *Symposium on Cloud Computing*, 2010.

[2] J. Chan, C. Hayes, and E. M. Daly. Decomposing discussion forums and boards using user roles. In *ICWSM*, 2010.

[3] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *OSDI*, pages 137–150, 2004.

[4] S. Ewen, S. Schelter, K. Tzoumas, D. Warneke, and V. Markl. Iterative parallel data processing with stratosphere: An inside look. *SIGMOD*, 2013.

[5] E. Friedman, P. Pawlowski, and J. Cieslewicz. Sql/mapreduce: a practical approach to self-describing, polymorphic, and parallelizable user-defined functions. *Proc. VLDB Endow.*, 2(2):1402–1413, 2009.

[6] F. Hueske, M. Peters, A. Krettek, M. Ringwald, K. Tzoumas, V. Markl, and J.-C. Freytag. Peeking into the Optimization of Data Flow Programs with MapReduce-style UDFs. In *ICDE*, 2013.

[7] M. Leich, J. Adamek, M. Schubotz, A. Heise, A. Rheinländer, and V. Markl. Applying Stratosphere for Big Data Analytics. In *BTW*, 2013.

[8] M. Rowe, M. Fernandez, S. Angeletou, and H. Alani. Community analysis through semantic rules and role composition derivation. *JWS*, 18(1), 2012.