Challenges in Dependable Internet-Scale Stream Processing

Peter Pietzuch
Department of Computing
Imperial College London
United Kingdom
prp@doc.ic.ac.uk

ABSTRACT

Today we lack an infrastructure for globally processing stream data from sensor networks and making this data available to millions of users in real-time. To build such a system, we need to address a set of challenges and, in particular, rethink what dependability means in this context: it is infeasible to guarantee perfect data processing at a global scale. Instead, the degradation of result quality due to failure and resource shortages should be made explicit to users. We briefly describe one such model to achieve a dependable Internet-scale stream processing service.

1. INTRODUCTION

With the commoditisation of embedded sensor devices, we witness an increasing number of deployments of wireless sensor networks around the world. Today, sensor networks exist in diverse areas, such as electronic health-care, environmental monitoring and urban transport management. It is safe to predict that their use will grow in the future.

Most deployed sensor networks share a common design, in which sensor nodes cooperate to deliver a stream of sensed data using radio communication to a base station node. The base station is often the only node in the system that has both wireless and wired network connectivity. Research deployments of sensor networks usually aim to successfully deliver sensed data to the base station node, without addressing any further data distribution and processing issues. This is not a complete solution because, in practise, users interested in the sensed data will not be located close to the sensor network. Instead, they may be scientists spread across research centres on different continents or thousands of web users sitting in front of their PCs.

The vision of a *global sensor web* attempts to address this issue with a world-wide infrastructure for interconnecting sensor networks [4]. Similar to how the web makes static content available, a global sensor web would allow millions of users to harness real-time stream data coming from different corners of the planet. Although the current Internet provides cheap connectivity, it was not built with this usage pattern in mind. Stream data requires reliable long-term transport between sensor networks and users and must be processed and aggregated while on transit in the network to reduce resource requirements.

We propose that a *dependable Internet-scale stream processing (DISSP)* system can make the global sensor web a reality. Similar to the ease of relational queries in DBMS, stream-processing systems allow users to access and manipulate distributed data streams through declarative queries. However, the scale of an Internet-wide system poses substantial challenges when it comes to providing a *dependable* service. Any such system must gracefully handle the failure of network links and processing hosts while managing a large pool of CPU and network resources.

To achieve this, we argue that DISSP needs a new model for dependability. This model cannot provide the hard guarantees of traditional DBMSs and today's stream-processing systems. Ensuring no tuple loss at all times may be feasible within a single data centre, but we cannot hope to achieve this at an Internet-scale. Instead, dependability should be driven by application requirements since many sensing applications can cope with a *controlled degradation* of result quality: a query supporting an earthquake monitoring application can accept the transient loss of data from a subset of sensor networks, as long as this leaves the probability of missing a seismic event negligible. We describe how such a dependability model can be realised by a DISSP system using an overlay network of nodes on today's Internet.

The rest of the paper is structured as follows: In §2, we derive requirements for DISSP from three application scenarios. We then describe the main research challenges that a DISSP system faces in §3. In §4, we present an approach for DISSP, focussing on dependability aspects. We finish with an overview of related work (§5) and conclusions (§6).

2. APPLICATION SCENARIOS

Diverse sensing applications fall under the DISSP model and can be supported by such a system. Applications are realised as declarative queries, resulting in fast deployment and resource sharing. Next we describe three representative applications and highlight their requirements.

2.1 Real-time astronomic data processing

The first application scenario ("*Astronomy*") provides a service to astronomers for correlating real-time image streams from geographically-distributed radio telescopes. The aim is two detect transient sky events, such as γ-ray bursts [22], in

WDDDM '08, March 31, 2008, Glasgow, UK

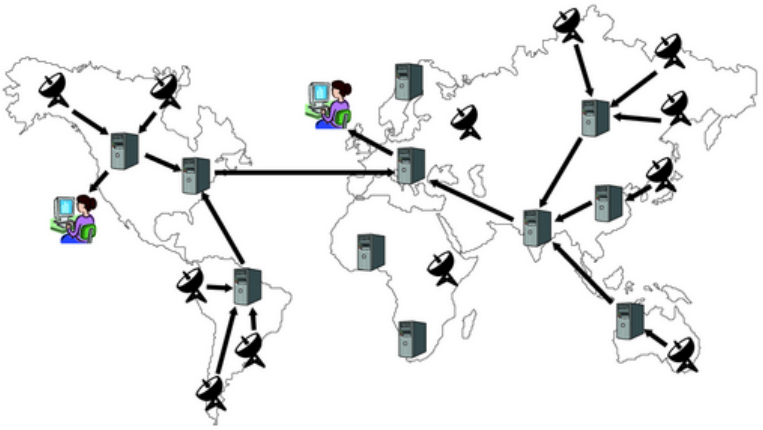

Figure 1: DISSP system as an overlay network, executing a query

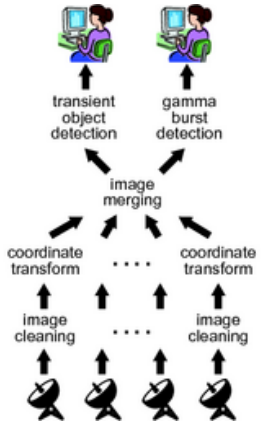

Figure 2: DISSP query for detection of transient sky objects

real-time. These events last for minutes and, after an event has been detected, instruments need to be re-aligned to focus on on-going occurrences.

Fig. 1 illustrates the deployment of a DISSP system, executing a query that takes images from radio telescopes, processes them in real-time and delivers data about transient anomalies to two astronomers. The processing is done by a distributed set of hosts (or data centres). The system must ensure that image data is transported reliably between processing sites. However, some data loss is acceptable, as long as no transient sky events are missed as a consequence.

The logical structure of the query implementing the application is shown in Fig. 2. The operators processing the image streams are specific to this application domain and cannot be expressed easily using, for example, relational query operators. There is also substantial overlap between the processing required by different astronomers who are interested in distinct types of anomalies.

2.2 Transportation infrastructure monitoring

In the second application scenario ("*Transport*"), we assume that the national transportation infrastructure, such as roads, tunnels and bridges, is instrumented with sensor networks to monitor their utilisation and condition [6]. Many parties run DISSP queries: the *road authority* monitors bridges and tunnels to get early warnings about critical damages; *traffic planers* issue queries about real-time traffic patterns across the road network, identifying hot-spots and aiding in the planing of new roads; and *web users* subscribe to traffic alerts affecting their daily commutes.

A challenge in this scenario is scalability in terms of query complexity and quantity. A single query such as "What is the road with highest utilisation between 6-7pm in the Greater London area?" may lead to the aggregation of stream data from thousands of roads. In addition, the number of live queries may be large if thousands of users submit queries about their commutes.

2.3 Global RSS feed mining (RSS)

The last application scenario ("*RSS*") looks at the processing of already existing stream data on the web: RSS feeds from websites and weblogs. Users want to apply data mining techniques across thousands of RSS feeds to uncover trends and correlations [20]: a *financial institution* is interested in real-time notifications about news-worthy events affecting financial markets; an *advertising agency* monitors weblogs to get an up-to-date picture about "hot" products discussed by bloggers; finally, *web users* receive custom RSS feeds with articles matching their interests about, say, the US election campaign. All of these queries require operators that implement natural language processing techniques. Such operators may have large computational requirements when parsing sentences of millions of news feed articles.

3. RESEARCH CHALLENGES

Next we highlight challenges in the area of DISSP, referring back to the above scenarios. The focus is on achieving dependability and we ignore other important issues such as security and privacy. The dependability challenges can only be solved through a combination of techniques from database, distributed systems and networking research.

3.1 Stream data and query models

An important decision for a DISSP system is the choice of data model. Sensor networks will produce data in different formats. Due to this heterogeneity, a single, unified data model for a global DISSP system may be unsuitable. Fortunately the database community has a long history of working on data heterogeneity and integration issues with directly applicable results to DISSP [4].

A DISSP system also needs a query model. As shown by the *Astronomy* and *Transport* scenarios, the query model should support custom processing operators. One possibility is to provide an API for the definition of new operators. However, query optimisers struggle to optimise queries with black box operators. A compromise may be to choose a restricted computational model for custom operators (*e.g.*, finite state automata). This supports non-trivial new operators, while allowing the query optimiser to reason about their semantics and resource requirements.

3.2 Dynamic data source discovery

Traditional DBMSs assume that users know about relevant input relations and include them in query expressions. This is infeasible in a DISSP system with many data sources: in the *RSS* scenario, the number of feeds scales with the number of websites. Therefore, data source discovery becomes a crucial requirement. Users describe their desired input data and the DISSP system maps this to a suitable set of sources. This mapping has to be dynamic because,

during the life-time of a query, new sources will join the system and old ones will disconnect or fail.

Data source discovery raises several challenges: (1) data sources must be able to describe the syntax and semantics of the data that they provide. This must be done intuitively to encourage the connection of new data sources; (2) the information about data sources must be disseminated effectively through the system; (3) data source discovery must be dynamic so that existing queries can be adapted.

3.3 Adaptive stream query optimisation

With many users running queries involving thousands of data sources, the DISSP system must allocate resources to queries: it must route query streams along network paths and map operators to processing hosts. This is similar to the task that a DBMS query optimiser carries out, however, there are differences resulting in new challenges: (1) the network is a limited resource that needs allocation. Often the performance of network paths will dominate the overall performance of queries; (2) due to the scale of DISSP, the query optimiser must work with incomplete knowledge about current system state. Resource availability of some nodes and network paths may be unknown; (3) load balancing and shedding become important features. In the *Transport* scenario, a simple aggregation query calculating the average traffic volume on all UK roads at a 1 Hz frequency could easily overload the system. The query optimiser must handle such query hot-spots; (4) due to the long-running nature of stream queries, optimisation must be adaptive. Failure and changes in network and host conditions will be common and require re-optimisation of query plans; (5) finally, the query optimiser should take advantage of the overlap between queries through multi-query optimisation. For example, in the *Astronomy* scenario queries for anomaly detection share the same data cleaning and transformation operators.

4. PROPOSED APPROACH FOR DISSP

In this section we describe how DISSP can achieve a reliable, fault-tolerant service. We assume that hosts located at distributed sites provide the processing infrastructure [19], as illustrated in Fig. 1. This deployment model has the advantage that resources are contributed through a federation of distinct administrative entities. Unlike in a general peer-to-peer model, hosts participating in the federation are dedicated and have a low churn rate. Grid deployments and PlanetLab, a distributed networking test-bed, are successful examples of such federated resource pools.

4.1 Quality-centric query model

We propose a *quality-centric* query model where a query is accompanied by a *utility function*. This function allows the DISSP system to calculate the user-perceived *quality value* of data tuples. Failure in the system will reduce data quality in a quantifiable way. For example, the utility of an aggregation query combining sky images from multiple telescopes may decrease proportionally with the number of unavailable telescopes due to the less precise results. By reducing *precision* to improve *availability* [18], a DISSP system guarantees a predictable stream processing service even with substantial resource failure.

Data quality metrics can be from two broad categories: **system-centric** metrics are independent of any given application and capture properties inherent to DISSP. Examples are *harvest* — the fraction of data sources with tuples included in the result stream of an aggregation query, *freshness* — the time until a user receives a tuple from a data source, and *throughput* — the rate at which tuples are delivered; **query-centric** metrics involve domain-specific knowledge about tuple semantics. Metrics such as *result accuracy*, *detection confidence* and *sample rate* fall into this category.

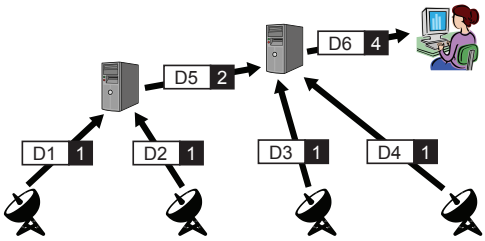

Figure 3: Propagation of quality-metrics with stream tuples

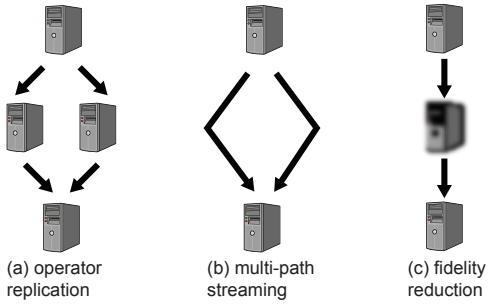

Figure 4: Fault-tolerance techniques in DISSP

As shown in Fig. 3, this query model can be implemented by associating data tuples in a stream with their quality values. In this example, the quality value is harvest and is updated as tuples traverse the query graph. For many applications, tuples closer to the user in the graph will have higher values — their loss has a larger impact because they represent aggregated data. The quality value may also decrease over time if freshness is part of the utility function.

4.2 Fault-tolerance techniques

A DISSP system can use different fault-tolerance techniques to maximise the utility of result tuples. As shown in Fig. 4, each technique rewrites the query plan into an equivalent representation: (a) **operator replication** adds redundant processing to the plan, preventing host failures from affecting results; (b) **multi-path streaming** adds redundant network paths to handle network failures; (c) **fidelity reduction** degrades processing quality, for example, by replacing exact operators with approximations or requesting lower grade streams from data sources. This reduces the resource requirements of a query so that the DISSP system can continue operation with diminished resources after failure.

The system must reconcile inconsistent state between replicated operators. Divergence may be caused by restarted failed hosts, temporary tuple loss due to network failure and network partitions. It is infeasible to withhold inconsistent results from users and replay corrected tuples because the system will lack sufficient resources to recover completely from missed processing. Instead, a DISSP system should accept a limited degree of inconsistency leading to a tem-

porary reduction of data quality. The system can reconcile inconsistent tuples according to several strategies: (a) select tuples using a **majority vote**; (b) select tuples with the **maximum quality value** as they are most useful to the user; (c) perform further processing to **combine tuples** into ones with higher data quality.

4.3 Strategies for stream query optimisation

A DISSP system must have a strategy for choosing among the above fault-tolerance techniques. In a system with thousands of queries, it is infeasible to expect a system administrator to manually allocate redundant resources for a DISSP service. The aim is to deliver result tuples with the highest quality to as many users as possible. This can be viewed as an optimisation problem: available resources are allocated to queries in order to maximise the overall user benefit under failure. More valuable tuples should be protected first. Some fault-tolerance techniques, such as operator replication, should be applied *proactively*, in anticipation of future failures, whereas others should be used *reactively* to ensure satisfactory performance of queries during failure, *e.g.,* by switching to approximate processing.

5. RELATED WORK

Global sensor web. The distributed systems community has recognised the need for global stream processing. [4, 9, 16, 17, 2]. However, previous efforts have either neglected dependability [9] or advocated reliability through substantial redundancy, thus wasting resources [10]. This is infeasible in an environment with resource-constraint hosts.

Distributed stream processing. Previous research has focused on moderate-scale, well-provisioned environments such as single data centres [1, 8, 5]. Therefore solutions for high availability mask failures, treating them as exceptions. The aim is to eventually deliver correct data to all users after a failure [3, 11, 21]. Such resource-intensive recovery limits scalability and is at odds with the ephemeral nature of real-time stream data. In addition, administrators have to manually configure redundant processing in anticipation of system failures. More recent work [13, 12] has considered deployments in wide-area networks but still assumes substantial over-provisioning of resources.

Grid. The Grid vision is to process large volumes of data collaboratively [7]. Most Grid applications follow a batch processing model, although researchers have begun looking at the needs of real-time streams [15, 14]. To achieve dependability, Grid applications are often built using redundant hardware and dedicated network links (*e.g., GridPP* and *TeraGrid*), leading to higher deployment costs. Instead, we advocate to exploit unreliable commodity hardware and shared networks for global stream processing.

6. CONCLUSIONS

By encouraging a fresh look at dependability in Internet-scale stream processing, we hope that such systems will play a crucial role in the future global sensor web. We also believe that these techniques for handling real-time stream data from sensor networks will provide useful input to efforts on next generation Internet designs. As a next step, we intend to validate our ideas with a public DISSP system on Planetlab. We will encourage sensor researchers to use our infrastructure to inter-connect real-world sensor networks.

7. REFERENCES

[1] D. J. Abadi, Y. Ahmad, et al. The Design of the Borealis Stream Processing Engine. In *CIDR*, 2005.

[2] K. Aberer, M. Hauswirth, and A. Salehi. Infrastructure for Data Processing in Large-scale Interconnected Sensor Networks. In *MDM*, May 2007.

[3] M. Balazinska, H. Balakrishnan, et al. Fault-Tolerance in the Borealis Distributed Stream Processing System. In *Proc. of SIGMOD'05*, Baltimore, MD, June 2005.

[4] M. Balazinska, A. Deshpande, et al. Data Management in the Worldwide Sensor Web. *IEEE Pervasive Computing*, 6(2):30–40, 2007.

[5] L. Chen, K. Reddy, and G. Agrawal. GATES: A Grid-Based Middleware for Processing Distributed Data Streams. In *Proc. of HPDC-13*, June 2004.

[6] EPSRC. WINES II Project - Smart Infrastructure. `winesinfrastructure.org`, 2006.

[7] I. Foster. What is the Grid? - A Three Point Checklist. *GRIDtoday*, 1(6), July 2002.

[8] M. Franklin, S. Jeffery, et al. Design Considerations for High Fan-in Sys.: The HiFi Approach. In *CIDR*, 2005.

[9] P. Gibbons, B. Karp, et al. IrisNet: An Architecture for a Worldwide Sensor Web. *Pervasive Computing, IEEE*, 2(4):22–33, Oct.-Dec. 2003.

[10] R. Huebsch, J. M. Hellerstein, et al. Querying the Internet with PIER. In *Proc. of VLDB*, Sept. 2003.

[11] J.-H. Hwang, M. Balazinska, et al. High-Availability Algorithms for Distributed Stream Processing. In *Proc. of ICDE'05*, Washington, DC, USA, 2005.

[12] J.-H. Hwang, U. Cetintemel, and S. Zdonik. Fast and Highly-Available Stream Processing over Wide Area Networks. In *Proc. of ICDE'08*, Cancun, Mexico, 2008.

[13] J.-H. Hwang, Y. Xing, U. Cetintemel, et al. A Cooperative, Self-Configuring High-Availability Solution for Stream Processing. In *Proc. of ICDE*, 2007.

[14] M. Ivanova and T. Risch. Customizable Parallel Execution of Scientific Stream Queries. In *Proc. of VLDB'05*, Trondheim, Norway, Aug. 2005.

[15] R. Kuntschke, T. Scholl, et al. Grid-Based Data Stream Processing in e-Science. In *Proc. of E-SCIENCE*, Washington, DC, USA, 2006.

[16] S. Midkiff. Internet-Scale Sensor Systems: Design and Policy. *Pervasive Comp.*, 2(4):10–13, Oct.-Dec. 2003.

[17] R. N. Murty and M. Welsh. Towards a Dependable Architecture for Internet-Scale Sensing. In *HotDep*, 2006.

[18] C. Olston and J. Widom. Offering a Precision-Performance Tradeoff for Aggregation Queries over Replicated Data. In *The VLDB Journal*, 2000.

[19] P. Pietzuch, J. Ledlie, J. Shneidman, M. Roussopoulos, M. Welsh, and M. Seltzer. Network-Aware Operator Placement for Stream-Processing Systems. In *Proc. of ICDE*, Apr. 2006.

[20] I. Rose, R. Murty, P. Pietzuch, et al. Cobra: Content-based Filtering and Aggregation of Blogs and RSS Feeds. In *Proc. of NSDI*, Cambridge, MA, USA, 2007.

[21] M. A. Shah, J. M. Hellerstein, and E. Brewer. Highly Available, Fault-tolerant, Parallel Dataflows. In *Proc. of SIGMOD*, New York, NY, USA, 2004.

[22] W. Vestrand, J. Wren, et al. Energy Input and Response from Prompt and Early Optical Afterglow Emmission in Gamma-Ray Bursts. *Nature*, 442, 2006.