<u>Exploiting Dynamic Resource Allocation for Efficient Parallel Data Processing in the Cloud</u>

ABSTRACT:

In recent years ad-hoc parallel data processing has emerged to be one of the killer applications for Infrastructure-as-a-Service (IaaS) clouds. Major Cloud computing companies have started to integrate frameworks for parallel data processing in their product portfolio, making it easy for customers to access these services and to deploy their programs. However, the processing frameworks which are currently used have been designed for static, homogeneous cluster setups and disregard the particular nature of a cloud. Consequently, the allocated compute resources may be inadequate for big parts of the submitted job and unnecessarily increase processing time and cost. In this paper we discuss the opportunities and challenges for efficient parallel data processing in clouds and present our research project. It is the first data processing framework to explicitly exploit the dynamic resource allocation offered by today's IaaS clouds for both, task scheduling and execution. Particular tasks of a processing job can be assigned to different types of virtual machines which are automatically instantiated and terminated during the job execution.

EXISTING SYSTEM:

A growing number of companies have to process huge amounts of data in a cost-efficient manner. Classic representatives for these companies are operators of Internet search engines. The vast amount of data they have to deal with every day has made traditional database solutions prohibitively
Expensive .Instead, these companies have popularized an architectural paradigm based on a large number of commodity servers. Problems like processing crawled documents or regenerating a web index are split into several independent subtasks, distributed among the available nodes, and computed in parallel.

PROPOSED SYSTEM:

In recent years a variety of systems to facilitate MTC has been developed. Although these systems typically share common goals (e.g. to hide issues of parallelism or fault tolerance), they aim at different fields of application. MapReduce is designed to run data analysis jobs on a large amount of data, which is expected to be stored across a large set of share-nothing commodity servers.

Once a user has fit his program into the required map and reduce pattern, the execution framework takes care of splitting the job into subtasks, distributing and executing them. A single Map Reduce job always consists of a distinct map and reduce program.

ALGORITHMS:

1. Job Scheduling and Execution:

After having received a valid Job Graph from the user, Nephele's Job Manager transforms it into a so-called Execution Graph. An Execution Graph is Nephele's primary data structure for scheduling and monitoring the execution of a Nephele job. Unlike the abstract Job Graph, the Execution Graph contains all the concrete information required to schedule and execute the received job on the cloud.

2. Parallelization and Scheduling Strategies:

If constructing an Execution Graph from a user's submitted Job Graph may leave different degrees of freedom to Nephele. The user provides any job annotation which contains more specific instructions we currently pursue simple default strategy: Each vertex of the Job Graph is transformed into one Execution Vertex. The default channel types are network channels. Each Execution Vertex is by default assigned to its own Execution Instance unless the user's annotations or other scheduling restrictions (e.g. the usage of in-memory channels) prohibit it.

MODULE DESCRIPTION:

1. NETWORK MODULE:

Server - Client computing or networking is a distributed application architecture that partitions tasks or workloads between service providers (servers) and service requesters, called clients. Often clients and servers operate over a computer network on separate hardware. A server machine is a high-performance host that is running one or more server programs which share its resources with clients. A client also shares any of its resources; Clients therefore initiate communication sessions with servers which await (listen to) incoming requests.

2. LBS SERVICES:

In particular, users are reluctant to use LBSs, since revealing their position may link to their identity. Even though a user may create a fake ID to access the service, her location alone may disclose her actual identity. Linking a position to an individual is possible by various means,

such as publicly available information city maps. When a user u wishes to pose a query, she sends her location to a trusted server, the anonymizer through a secure connection (SSL). The latter obfuscates her location, replacing it with an anonymizing spatial region (ASR) that encloses u. The ASR is then forwarded to the LS. Ignoring where exactly u is, the LS retrieves (and reports to the AZ) a candidate set (CS) that is guaranteed to contain the query results for any possible user location inside the ASR. The AZ receives the CS and reports to u the subset of candidates that corresponds to her original query.

3. SYSTEM MODEL:

The ASR construction at the anonymization process abides by the user's privacy requirements. Particularly, specified an anonymity degree K by u, the ASR satisfies two properties: (i) it contains u and at least another K * 1 users, and (ii) even if the LS knew the exact locations of all users in the system.

- We propose an edge ordering anonymization approach for users in road networks, which guarantees K-anonymity under the strict reciprocity requirement (described later).
- We identify the crucial concept of border nodes, an important indicator of the CS size and of the query processing cost at the LS.
- We consider various edge orderings, and qualitatively assess their query performance based on border nodes.
- We design efficient query processing mechanisms that exploit existing network database infrastructure, and guarantee CS inclusiveness and minimality. Furthermore, they apply to various network storage schemes.
- We devise batch execution techniques for anonymous queries that significantly reduce the overhead of the LS by computation sharing.

4. SCHEDULED TASK:

Recently, considerable research interest has focused on preventing identity inference in location-based services. Proposing spatial cloaking techniques. In the following, we describe existing techniques for ASR computation (at the AZ) and query processing (at the LS). At the end, we cover alternative location privacy approaches and discuss why they are inappropriate to our problem setting. This offers privacy protection in the sense that the actual user position u cannot be distinguished from others in the ASR, even when malicious LS is equipped/advanced enough to possess all user locations. This spatial K-anonymity model is most widely used in location privacy research/applications, even though alternative models are emerging.

5. QUERY PROCESSING:

Processing is based on implementation of the theorem uses (network-based) search operations as off the shelf building blocks. Thus, the NAP query evaluation methodology is readily deployable on existing systems, and can be easily adapted to different network storage schemes. In this case, the queries are evaluated in a batch. we propose the network-based anonymization and processing (NAP) framework, the first system for K-anonymous query processing in road networks. NAP relies on a global user ordering and bucketization that satisfies reciprocity and guarantees K-anonymity. We identify the ordering characteristics that affect subsequent processing, and qualitatively compare alternatives. Then, we propose query evaluation techniques that exploit these characteristics. In addition to user privacy, NAP achieves low computational and communication costs, and quick responses overall. It is readily deployable, requiring only basic network operations.

HARDWARE & SOFTWARE REQUIREMENTS:

HARDWARE REQUIREMENTS:

• System	:	Pentium IV 2.4 GHz.
• Hard Disk	:	40 GB.
• Floppy Drive	:	1.44 Mb.
• Monitor	:	15 VGA Colour.
• Mouse	:	Logitech.
• Ram	:	512 MB.

SOFTWARE REQUIREMENTS:

• Operating system	:	Windows XP Professional.
• Coding Language	:	ASP .Net,C#
• Database	:	Sql Server 2005.

REFERENCE:

Daniel Warneke and Odej Kao, "Exploiting Dynamic Resource Allocation for Efficient Parallel Data Processing in the Cloud", **IEEE Transactions on Parallel and Distributed Systems, January 2011.**