T. Grust et al. (Hrsg.): Datenbanksysteme für Business, Technologie und Web (BTW 2019),
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2019 537

NeMeSys — Energy Adaptive Graph Pattern Matching on NUMA-based Multiprocessor Systems

Alexander Krause[1] Annett Ungethuem[1] Thomas Kissinger[1] Dirk Habich[1] Wolfgang Lehner[1]

Abstract: NeMeSys is a NUMA-aware graph pattern processing engine, which leverages intelligent resource management for energy adaptive processing. With modern server systems incorporating an increasing amount of main memory, we can store graphs and compute analytical graph algorithms like graph pattern matching completely in-memory. Such server systems usually contain several powerful multiprocessors, which come with a high demand for energy. We demonstrate, that graph patterns can be processed in given performance constraints while saving energy, which would be wasted without proper controlling.

Keywords: Graph, Pattern Matching, NUMA, Adaptivity, Energy

1 Introduction

Graph pattern matching is an important, declarative, topology-based querying mechanism and a core primitive in graph analysis. Fundamentally, graph pattern matching is important to many applications such as analyzing hyper-links in the World Wide Web, fraud detection, biomolecular engineering, scientific computing, or social network analytics, only to name a few, as in Krause et al. [Kr17]. The query pattern is usually given as a graph-shaped pattern and the result is a set of matching subgraphs.

On the one hand, the calculation of graph patterns can get prohibitively expensive, because of a possibly high number of intermediate results. On the other hand, modern hardware systems feature main memory capacities of several terabytes, so that we are able to store and process graphs entirely in main memory. Based on that, we have built NeMeSys, a near memory graph pattern processing system being built upon the *Data-Oriented Architecture (DORA)* as used in Kissinger et al. [KHL18] to satisfy high performance demands in an efficient way. However, modern scale-up server systems usually contain several multiprocessors consuming high amounts of energy during data processing. Unfortunately, the energy efficiency of a graph pattern matching system is often not considered, because of the high performance demand for the pattern matching process.

[1] Technische Universität Dresden, Database Systems Group, Noethnitzer Straße 46, 01187 Dresden,
<firstname>.<lastname>@tu-dresden.de

doi:10.18420/btw2019-40

To tackle that issue, we want to demonstrate at the BTW conference, that the energy control findings of Kissinger et al. [KHL18] for relational systems can be also transferred to the substantially more complex graph context. In our demo, we want to emphasize the importance of energy control and the negligible performance impact by using intelligent energy control loops.

2 System Description

From a hardware perspective, the scale-up approach is mainly characterized by the fact that separate memory domains per processor are implemented which are remotely accessible via an interconnect network resulting in a *non-uniform memory access (NUMA)* behavior. To tackle the limiting issues of increased latency and decreased bandwidth when accessing remote memory domains as shown by Kissinger et al. [KHL18], NeMeSys leverages well-known DORA and *near-memory computing (NMC)* principles with its basic architecture as portrayed in Figure 1. NMC means, that we limit the scope of each worker to memory domains, which are directly connected to their socket (local instead of remote accesses). Moreover, the DORA approach defines, that only one worker at a time is allowed to touch a certain data partition. This architectural decision implicates data partitioning, which we apply following our guidelines from Krause et al. [Kr17]. During query processing, intermediate states are communicated between the workers with both local and inter-socket messages via the messaging interface, which fetches the correct receiver from the partition manager (cf. Figure 1). To tackle the energy adaptivity, NeMeSys reuses the *Energy Control Loop* approach from Kissinger et al. [KHL18], which constantly monitors system utilization and applies appropriate core configurations.

In NeMeSys, graph queries can be entered through a user interface, which are then parsed by a query compiler. Every query consists of a chained set of operators, which are derived from the given query pattern. When the query gets executed, every worker in the system

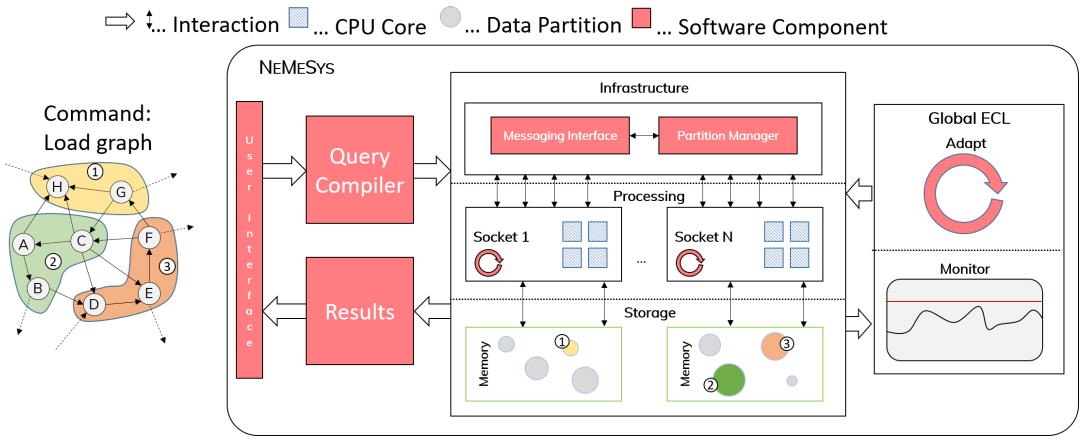

Fig. 1: NeMeSys system design

Fig. 2: Operating system governed processing

Fig. 3: Full Demo UI with energy controlled processing

can fork the same operator code and apply it to any data partition on its socket to achieve maximal parallelism. Therefore high data locality is important for minimal inter-socket messages and thus less energy overhead for the network communication. An operator will forward its intermediate matching state to the next operator in the chain, until all messages in the system have been processed.

3 Demo Booth

We show the adaptivity of NeMeSys based on Wikidata[2], Wikipedias underlying knowledge graph. In contrast to Ungethüm et al. [Un17], our use case imposes increased complexity because of the NP hardness of graph pattern matching and thus demands for more sophisticated execution and adaption mechanisms. Our dataset is based on a filtered truthy statement dump[3] with roughly 224 M edges. We generated our query set based on the anonymized Wikidata SPARQL query logs[4]. From these logs, we created three different workload profiles for daily, hourly and minute wise query load fluctuation. The used query

[2] https://www.wikidata.org/

[3] https://dumps.wikimedia.org/wikidatawiki/entities/

[4] https://iccl.inf.tu-dresden.de/web/Wikidata_SPARQL_Logs/en

set contains approximately 1.2 M different graph pattern and arbitrarily length path queries with sizes ranging between 1 and 52 edges per query.

During the demo, the user can change the load profile and switch between using our energy control loop or letting the operating system take over with the controls at (1) in Figure 3. Our front end shows the arriving queries per second (2) and the graph at (3) displays the systems power draw over time. Switching between the operating system and our controlling mechanisms yields different energy consumption as shown in Figures 2 compared to Figure 3. One of the core components is the chart at (4). It displays the number of active cores per socket and color codes the currently configured frequency per core and socket, where grey means a sleeping CPU, green means a low frequency, yellow medium scaled frequency and red resides in the highest frequencies for that core or socket. The four charts at (5) the Work-Energy Profiles of Ungethüm et al. [Un17] are used to allow for a fine grained system configuration. The system chooses one configuration, such that the queued tasks do not exceed a performance threshold, which we defined as 1 s for this experiment, as shown at (6) with the task counter and speedometer.

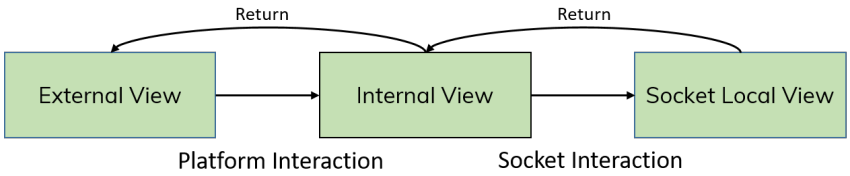

Fig. 4: Virtual Reality interaction guide

Our UI is able to display the most interesting information, however, it lacks the support of understanding the actual underlying hardware architecture. Thus, we visualize the system in an abstracted Virtual Reality (VR) environment, which the user can explore using VR glasses. Demo visitors can interact with the objects in the scene, e.g. dive into the server and select a specific component to display relevant related information. Figure 4 shows the possible user interactions to change scenes. The external view show meta information about the current experiment. Interacting with the shown platform leads the user to a new scene with visualized sockets and interconnects, where the user can choose to display statistics about the elements. The socket local can be reached via interacting with the sockets and yields per-core information.

References

[KHL18] Kissinger, T.; Habich, D.; Lehner, W.: Adaptive Energy-Control for In-Memory Database Systems. In: SIGMOD. Pp. 351–364, 2018.

[Kr17] Krause, A.; Kissinger, T.; Habich, D.; Voigt, H.; Lehner, W.: Partitioning Strategy Selection for In-Memory Graph Pattern Matching on Multiprocessor Systems. In: Euro-Par. Pp. 149–163, 2017.

[Un17] Ungethüm, A.; Kissinger, T.; Mentzel, W.; Mier, E.; Habich, D.; Lehner, W.: Energy Elasticity on Heterogeneous Hardware using Adaptive Resource Reconfiguration. In: BTW. P. 615, 2017.