On-the-Fly Data Synopses: Efficient Data Exploration in the Simulation Sciences

Thomas Heinis[†], David A. Ham[‡]

[†]*Department of Computing* [‡]*Department of Mathematics*

Imperial College, London, UK

ABSTRACT

As a consequence of ever more powerful computing hardware and increasingly precise instruments, our capacity to produce scientific data by far outpaces our ability to efficiently store and analyse it. Few of today's tools to analyse scientific data are able to handle the deluge captured by instruments or generated by supercomputers.

In many scenarios, however, it suffices to analyse a small subset of the data in detail. What scientists analysing the data consequently need are efficient means to explore the full dataset using approximate query results and to identify the subsets of interest. Once found, interesting areas can still be scrutinised using a precise, but also more time-consuming analysis. Data synopses fit the bill as they provide fast (but approximate) query execution on massive amounts of data. Generating data synopses after the data is stored, however, requires us to analyse all the data again, and is thus inefficient.

What we propose is to generate the synopsis for simulation applications on-the-fly when the data is captured. Doing so typically means changing the simulation or data capturing code and is tedious and typically just a one-off solution that is not generally applicable. In contrast, our vision gives scientists a high-level language and the infrastructure needed to generate code that creates data synopses on-the-fly, as the simulation runs. In this paper we discuss the data management challenges associated with our approach.

1. INTRODUCTION

Over the last half century, computer simulations of natural phenomena have become an indispensable part of the scientific method [2, 17]. For the study of many natural phenomena, building a model and simulating it complements the understanding developed using traditional methods. In fields as diverse as cosmology, seismology and climate science, simulation is essential in understanding phenomena when a physical experiment is impossible.

The models used today are as big and detailed as the memory of the simulation infrastructure allows, and with every successive hardware generation, the models grow larger. As the computing infrastructure evolves, the amount of data generated increases. Even today, datasets may be as large as petabytes and are growing rapidly [10], making them difficult to store.

Storage capacity, however, is only one aspect of the problem. The data generated by simulations is so big that the analysis of the data is also a severe challenge. Complete analysis, however, is rarely required and what scientists need instead is explorative access to the data to find interesting subsets that need further detailed and time-consuming analysis. Data synopses providing approximate (with a tight bound for the maximum error) but fast query results are a promising first step to allow for the scalable exploration of simulation results.

Creating the synopses after the simulation output has been written to disk, however, means revisiting and analysing all the data; a time-consuming process (multiple times depending on the synopsis). We consequently propose to develop a broadly applicable approach through generating code for creating the synopsis on-the-fly, according to the needs of the scientists and optimised for the architecture used. This would leverage and complement recent work [11, 12] on the development of high-level languages to generate low-level, hardware-optimised simulation code. We envision a system in which, given the simulation code and model, the user describes the architecture and the questions they are interested in and the system generates the code to produce this exact synopsis as the simulation runs.

To build such a system, we need to address several challenges. First, we need to define a language the user can use to express their interest. Second, depending on the user's interests, a particular type of synopsis must be chosen. For example, if a user needs to ask for simple range or count queries, a synopsis based on histograms may be sufficient. Third, the generation of the synopsis needs to be fast and thus optimised for the hardware architecture used.

This work was supported by the Natural Environment Research Council [NERC grant reference number NE/K008951/1].

Finally, to make the best use of the space needed to store the synopsis, we need synopses which are more accurate in areas that scientists are interested in and less so elsewhere.

Explorative queries on scientific data [9] (potentially relying on model-based compression [14]) have already been proposed in the past. Our vision applies similar ideas of approximate query answers (albeit not with iterative, i.e., with improving precision [9, 1]) to the simulation sciences to automatically produce data synopses. Work on reasoning about approximate and uncertain data [6] ideally complements our work in that it can be used to reason about approximate query answers.

2. BACKGROUND

The vision we propose touches on three different areas of research.

2.1 Simulation Sciences

Over the last half century, computer simulation of physical phenomena has joined theory and experiment to form a third, and indispensable, pillar of scientific research [17]. For the vast range of physical phenomena which are continuous, the simulations centre on the calculation of discrete approximate solutions to partial differential equations. The spatial extent of the simulation is typically modelled by a finite vector of solution values, connected by a grid or mesh structure. The scientist specifies an initial simulation state, and the simulation state at the next discrete time step is calculated by applying numerical operations.

Scientists constantly push the precision, size and duration of simulations to the limits of their computational infrastructure. The size of the simulation is typically limited by the CPU power, speed of data transfer and, size of main memory in which the current state is stored.

The massive simulation result that has to be analysed considerably challenges the state of the art in data analysis as most current approaches do not scale well. One of the biggest challenges in simulation science today is indeed the size of the simulation result. While ever bigger disks enable simulation results of increasing size to be stored, the lack of scalable analysis tools limits the simulation duration in many instances [13]: why simulate for longer if the results cannot be analysed?

2.2 Generating Simulation Code

Numerical methods and simulation hardware have become so sophisticated in recent years that it takes scientists enormous levels of effort and very specialised computational skills to take advantage of them [12]. They frequently spend months or years reinventing implementations for numerical methods that are not optimal for the underlying highly parallel hardware. To make matters worse, the resulting code is often hundreds of thousands of lines, making it very difficult and time-consuming to maintain in the face of ever more sophisticated algorithms designed for ever more complex computer hardware.

Recent advances, however, have introduced higher-level languages to formulate the simulation problem on a mathematical level with only a few hundreds of lines of code [11, 12]. The mathematical specification is used to automatically generate highly optimised code for the hardware it is run on. Using a high-level language combined with code generation creates a critical separation of concerns: the scientist specifies the numerical algorithm but not its implementation. It is the role of the code generation system to produce a high performance parallel implementation highly optimised for the underlying hardware platform.

The Unified Form Language (UFL), which is employed by both the FEniCS [11] and Firedrake [12] automated simulation packages, enables scientists to work at this very high level, specifying mathematical operations over data stored in a wide range of representations on a computational mesh.

2.3 Data Synopses

Research in data synopses has in the past primarily been driven by applications like data streams and cardinality estimation for query processing [5]. In their very nature, however, they present a great opportunity to accelerate approximate query execution in the context of big data.

The basic idea of data synopses is that the full dataset is summarised, typically by using compression [5]. The synopsis acts as a surrogate for the data and is queried instead of the complete dataset. Through compression or summarisation, the synopsis is usually considerably smaller than the full dataset itself and consequently queries are executed faster. The execution time of a query depends primarily on the size of the synopsis. The size of the synopsis in turn depends on dataset characteristics, for example how compressible the data is, and is further controlled by the level of the compression.

Because of its substantial compression ratios, lossy compression is often used leading to imprecise representations of the data. Data synopses based on lossy compression can thus only answer queries approximately. Clearly, there is a trade-off between the size of the data synopsis (and thus query execution time) and the quality of the approximation: the smaller the synopsis is, the less accurate the approximation is and thus the bigger the error. Regardless of the quality of approximation used, the key to data synopses is to provide a user with tight

error bounds expressing how accurate the result is.

Data synopses have in the past primarily been used for data streams and cardinality estimation (for query planning) [5]. With data growing beyond what can be today handled efficiently, however, data synopses are rediscovered and are considered an interesting approach: instead of analysing the potential petabytes in big data applications in a time-consuming process, substantially smaller synopses can be queried almost instantly.

One major obstacle hindering adoption of data synopses is that the process to build them is time-consuming. Only once the experiment data is written to disk, are the synopses computed. All data, terabytes or more, has to be (potentially repeatedly) again read from the disk, loaded into memory, processed and written to disk. Having researchers wait until the synopsis is built before they can analyse the data considerably limits productivity [13].

Considerable effort in the past has primarily developed four types of synopses. First, random sampling [15] is very well suited for aggregate queries and takes samples at random either out of the results of the running simulation as they are streamed from the supercomputer to disk or after the dataset is stored on disk after the simulation. The former, sampling as the simulation runs, is very efficient, but requires all potential queries to be known at runtime already. For the latter, in case the dataset is stored on disk, samples can be taken online, at query time from the full dataset, or offline. For massive data, however, taking the samples online from the full dataset, as the query is asked, is unlikely to be feasible due to the high cost of random access to disk. Instead, sampling for big data needs to take samples offline. A second well-researched type of synopses are histograms [8] which play a central role for cardinality estimation in databases. Histograms summarise the data into bins, each with its own value range, e.g., the bins store count of values/tuples in its range. Doing so makes them particularly useful for range-count queries, but they are also used for general analysis queries [5]. Synopses based on wavelets summarise and approximate the data through wavelets [3]. Essentially, wavelet transformation is applied to relations or to time series, resulting in a collection of wavelet coefficients. The size of the synopsis depends on how many coefficients are stored, which in turn defines the accuracy with which queries can be answered. The size of the synopsis, however, alone does not define the query execution time: at runtime, query execution can choose to ignore coefficients, thereby reducing query execution time, but also precision. Synopses based on sketches are a relatively new development [4]. As opposed to sampling, all data is considered for sketches, but only a small summary is retained (e.g., for a sum query all values are added up and only the sum is stored). A different sketch has to be defined for each query and this approach thus requires considerable effort.

3. GENERATING SYNOPSES

At the core of our approach is the idea of using data synopses for the efficient exploration of data produced as the result of the simulation. Producing the synopses after the data has been written on disk means revisiting all the data again to compute the and is therefore costly.

3.1 Core Idea

Instead, we propose to instrument the model code to generate the synopsis in a unified mesh and vector data model during the simulation. Given a space budget and, potentially, priorities specified by the scientists, the system will select an appropriate synopsis representation and resolution. This will enable scientists to efficiently analyse, query and explore the simulation result approximately (with a tight maximum error bound) as soon as the data is output: possibly even while the simulation still continues. Scientists will specify queries in a high level mathematical manner and the required query code will be generated automatically. Scientists can use the synopsis with or without the full dataset (and writing all data to disk may be avoided). In some cases the synopsis alone already allows for a detailed enough analysis whereas in other instances, the synopsis serves as a surrogate for finding interesting phenomena that are then analysed in the complete dataset.

As we will discuss later, the choice of the type of synopses heavily depends on the type hypothesis to be validated through the simulation. As a consequence, we further propose that scientists formulate queries (based on the simulation model) they wish to answer using the simulation results.

The queries can additionally help to improve the usefulness of the synopsis. Given a limited space budget, the scientist may also be able to indicate areas or time ranges of interest *a priori*. We thus propose to generate synopses with variable precision, i.e., having a higher precision in areas of interest and lower precision elsewhere. Higher precision translates to smaller error bounds of the approximate answers but also requires more space. Clearly, it is not wise to only focus on this range as other queries could no longer be answered, so variable precision introduces an interesting trade-off in the face of a limited space budget. The queries serve as a crucial hint to generate a synopsis with more details in areas of the data where scientists want to query.

To optimise the generation of the synopsis, we

also require information about the underlying hardware. Information about the hardware is used as input to the code generation, so that computing and storing the synopsis as well as the result can be optimised for the underlying hardware. Doing so can optimise use of limited computation power and bandwidth to storage devices, therefore ultimately reducing interference with the running simulation.

Ultimately our goal is to generate the code to efficiently compute and store a synopsis of the simulation results on disk, along with code for the simulation itself. The synopsis written is of variable precision and the code generation is based on a space budget, the simulation model, the user queries used as hints, as well as information about the underlying hardware. The resulting code efficiently generates a data synopsis that will enable scientists to quickly and scalably analyse simulation results, without having to tinker with code or deal with the peculiarities of data synopses and hardware.

While similar ideas could be used for observation sciences as well, doing so is difficult in practice. Although pushing filters deep into the data acquisition pipeline is already standard practice so that events of little interest are filtered close to the detector, the instruments in the observational sciences are custom built, making code generation challenging.

3.2 Application Example

To better understand the brain and develop new drugs for brain-related diseases, the scientists in the Human Brain Project [13] build small-scale bio-realistic spatial models of a rat brain to simulate them in-silico. The spatial models they design are based on millions of three-dimensional cylinders, where several thousand cylinders together reconstruct the spatial shape of one neuron.

In their simulations on a supercomputer (BlueGene/Q) they study the propagation of electrical impulses through the models. At every time step of the simulation, the voltage (or other parameters) at each cylinder of every neuron (out of the billions in the model) is measured. The simulation output is essentially the state over the course of the simulation, i.e., one time series per cylinder of the model.

The resulting datasets are oftentimes of unprecedented size but do not need to be examined in detail in a first instance. Instead, to understand the simulation result, multiple queries of an explorative nature are asked. An example of a frequent post-simulation analysis used to understand the simulation data is to determine in what areas of the model the measured parameter, e.g., voltage, exceeds a particular threshold. To find such events, the neuroscientists execute queries with a combination of spatial, temporal and voltage predicates. Analyses based on the simulation model in general follow this template, i.e., restricting the temporal, spatial as well as voltage values to particular ranges.

Data synopses generated on-the-fly are ideal for this application scenario. Data synopses can be generated instead of the simulation data and allow for a substantially faster approximate analysis and exploration. Alternatively, in case the synopsis is generated along with the full data, it can be used to find interesting areas while the detailed analysis can be performed on the identified subsets in the full data.

Researchers in general typically carry out experiments with assumptions or expectations and consequently are predominantly interested in particular ranges of the values (areas, temperature ranges, etc.). This information can be based on previous analyses or on assumptions by the researchers but is crucial, as the synopsis can be more precise in areas of interest and more approximate in others. This is no different in the Human Brain Project, where researchers may be interested in finding particular events (shapes of the voltage curve) in a voltage range between 10 and 23. Clearly, a higher precision of the synopsis in this range is helpful for them.

Writing such synopses to permanent storage is challenging as the underlying simulation hardware typically features novel and cutting edge hardware components. The BlueGene/Q used in the Human Brain Project, for example, features SSD storage that can be used to buffer the simulation output. Doing so is essential when generating synopses as the simulation results can be buffered and summarised/consolidated (using the CPUs of the SSD racks) on-the-fly as the simulation is running, before writing the synopsis to the disk.

4. RESEARCH CHALLENGES

Generating code for on the fly synopses entails interesting data management challenges around synopses themselves (choice of adequate type as well as synopses with variable precision), the language to express user queries used as hints and finally optimising their computation used for the hardware. We discuss each of the challenges in the following.

4.1 Data Synopses

The research challenges around synopses centre around deciding what type of synopsis to use, as well as how to configure it. More challenging is the how to support variable precision in data synopses.

4.1.1 Data Synopsis Type and Configuration

One research question that needs to be addressed is the choice of data synopses. Most types of synopsis developed in the past primarily focus on answering aggregate queries. Histograms [8] are ideal for aggregates while sampling [15], wavelets [3] or sketches [4] can be used for more general queries.

It is therefore key to understand, based on the sample queries the user provides, what questions need to be answered and what type of synopsis is suitable for these queries. If the sample queries hint at aggregate queries, a synopsis based on histograms can be used. Also based on the queries is the decision on how to configure the synopsis. In the case of histograms the question is what statistics are stored per bin (e.g., for maximum predicates, each bucket must store the maximum value). If sketches are used, the example queries indicate what information the sketches need to capture.

In the context of the Human Brain Project (HBP), the simulation output primarily comprises of time series measuring the voltage in different areas of the brain. The queries asked for the most part are threshold queries, e.g., in what time series does the voltage exceed a given threshold. For this type of queries wavelets may be most suitable.

4.1.2 *Synopsis with Variable Precision*

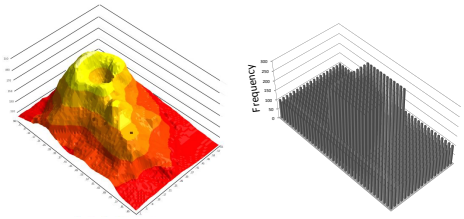
Key to the vision of generating synopses on-the-fly is to use variable precision or resolution. Using variable resolution allows us to reduce the space required by the synopsis, as areas which are not of much interest can be represented with less data.

The sample queries are used to infer what ranges (e.g., areas in a spatial model) the scientist is interested in exploring and analysing. The resolution is set higher for areas the scientist is interested in, and lower elsewhere. Improving the precision in areas of interest can be accomplished by storing more samples in the case of sampling, storing more coefficients in the case of wavelets or by using smaller bins in the case of histograms.

Figure 1 illustrates a synopsis with variable precision. The left hand side shows a spatial model (of a volcano with the temperature plotted) where the scientists are primarily interested in the center. As consequence, the center of the model is sampled with a higher frequency to produce a synopsis based on sampling as is shown on the right hand side (the y-axis shows sampling frequency while the other axes show geographic location).

In the sample application of the HBP, the neuroscientists primarily want to study voltage spikes and consequently they need higher resolution in ranges where spikes occur. Similarly, they are frequently interested in a particular spatial area. Only simulating the area of the model they are interested in is not feasible as the behaviour of the entire model must be simulated to obtain correct results. Specifying what areas to store with high precision is key to use space and bandwidth efficiently.

Using variable precision, however, introduces several challenges. First, given a space budget for the synopsis, the question is how much space should be used for ranges that the scientist is interested in, and how much should be used for the remaining areas. Using all space for the ranges of the areas of interest, will make analysis of other areas outright impossible and an adequate trade-off needs to be found. Second, query answers are straightforward to compute if the query only touches parts of the synopsis with the same precision. New approximate query answering approaches, however, need to be developed for the case where the precision is mixed. Third, crucial to this idea is that the increased precision is also quantifiable. That is, in areas with high precision, the error bounds are tighter compared to other areas. New methods have to be developed to quantify the error bounds in case ranges with varying precision are used.



(a) Spatial Model (b) Sampling Freq.

Figure 1: Sampling with higher frequency in areas of interest.

4.2 Model Definition & Query Language

The critical insight which makes the code generation approach feasible is that a wide range of simulation software employs variations of the same mesh and vector data model. Mathematical queries written in UFL can therefore be mapped to the data of models which are not themselves written in UFL. In addition to UFL, however, it will be necessary to define a suitable spatial query language. This will enable the scientist to specify the area or points over which the query should operate. This language is the layer which must be mapped to a database query to efficiently extract the relevant data.

4.3 Hardware Optimizations

Taking into account the available hardware can yield considerable improvements when computing and writing the simulation synopsis (as well as the simulation result) to permanent storage. Writing the synopsis efficiently in the face of limited disk bandwidth is crucial, as is exploiting special purpose hardware to offload computation of the simulation result so as not to interfere with the computation of the simulation result.

Supercomputers used for simulations frequently only have an aggregated disk bandwidth of a few tens of Gb/s for sequential I/O shared among sev-

eral thousand cores. A BlueGene/Q deployment, for example, has 16K cores sharing 40Gb/s of bandwidth, creating a serious bottleneck. Given the limited bandwidth, individual cores cannot simply write data to disk whenever they have new simulation results or new data synopses information. Instead, data has to be buffered at the cores and written to disk orchestrated, thereby wasting main memory that could be used to store bigger models.

Novel hardware solutions to alleviate this issue are at the ready. Active, flash-based storage [16] can be used to a) buffer bursts of simulation results, and b) use spare cycles of the active storage to compute the synopses. As a consequence, the simulation results are written on external flash drives where their synopses can be computed on-the-fly, thereby also offloading the computational overhead of computing the synopses.

Graphics Processing Units (GPUs) are becoming increasingly prevalent in the simulation sciences where they are used to offload parallel computations. The data transfer needed to move computation from the CPU to the GPU and simulation results back again, however, is a bottleneck because of the limited bandwidth of the PCI bus [7]. Data has to be accumulated so it can be transferred in a batch and incurs as little overhead as possible. Also in this scenario, the question is where to compute the synopsis and how to transfer it to the CPU where it can be stored permanently.

Crucially, the hardware has to be considered in order to optimise when and where to compute the synopses and when to write it to disk. Optimising use of the hardware, however, is beyond the abilities of a simulation scientist and is a challenge for which the code is ideally generated as well. The research challenge consequently is to determine how to describe the hardware infrastructure and how to generate code that uses efficient strategies to compute and store data synopses. The HBP, for example, has flash-based storage to buffer simulation results. Its efficient use, however, is beyond the abilities of simulation scientists and using code generation to develop code for producing synopses efficiently on the active storage is crucial.

5. CONCLUSIONS & FURTHER IDEAS

Simulations produce massive amounts of data that are difficult to analyse efficiently. What limits simulation size and length today is not only disk space, but our capacity to analyse the data within a reasonable time. Data synopses offer an interesting approach to the problem of the data deluge resulting from simulations. They are, however, challenging to compute and use, particularly for domain scientists with little training in software development.

With this vision, we propose an approach that avoids domain scientists having to develop highly specialised code to write data synopses. Instead, the code is automatically generated based on a space budget for the synopsis, the simulation model, example queries and a specification of the hardware used. The synopses enable scientists to scalably analyse massive data.

An interesting future avenue for this research is to generate optimised code for indexes as well. Clearly, each particular simulation will need different types of indexes but with sample analysis queries we can potentially infer what indexes are required.

A further interesting research question is if and how queries can be answered based on the static analysis or symbolic execution of the simulation code. Clearly, randomness is inherent in scientific simulations and the results will depend on the input parameters but assumptions about either can be made to potentially reason about query results without executing the simulation.

6. REFERENCES

[1] S. Agarwal, B. Mozafari, and et al. BlinkDB: Queries with Bounded Errors and Bounded Response Times on Very Large Data. EuroSys, 2013.

[2] J. L. Casti. *Would-be Worlds: How Simulation is Changing the Frontiers of Science*. Springer, 1996.

[3] K. Chakrabarti, M. N. Garofalakis, R. Rastogi, and K. Shim. Approximate Query Processing Using Wavelets. In *VLDB '06*.

[4] G. Cormode and M. Garofalakis. Sketching Streams Through the Net: Distributed Approximate Query Tracking. In *VLDB '05*.

[5] G. Cormode, M. Garofalakis, P. J. Haas, and C. Jermaine. Synopses for Massive Data: Samples, Histograms, Wavelets, Sketches. *Found. Trends databases*, 4(1):1–294.

[6] B. Gonçalves and F. Porto. ϒ-DB: Managing Scientific Hypotheses as Uncertain Data. *VLDB*, 2014.

[7] C. Gregg and K. Hazelwood. Where is the Data? Why You Cannot Debate CPU vs. GPU Performance Without the Answer. ISPASS '11.

[8] Y. Ioannidis. The History of Histograms (Abridged). In *VLDB '05*.

[9] M. L. Kersten, S. Idreos, S. Manegold, and E. Liarou. The Researcher's Guide to the Data Deluge: Querying a Scientific Database in Just a Few Seconds. *VLDB*, 2011.

[10] B. Lawrence, V. Bennett, J. Churchill, M. Juckes, P. Kershaw, S. Pascoe, S. Pepler, M. Pritchard, and A. Stephens. Storing and manipulating environmental big data with JASMIN. In *Proceedings of the IEEE International Conference on Big Data,*, 2013.

[11] A. Logg, K.-A. Mardal, G. N. Wells, et al. *Automated Solution of Differential Equations by the Finite Element Method*. Springer, 2012.

[12] G. R. Markall, D. A. Ham, and P. H. J. Kelly. Towards Generating Optimised Finite Element Solvers for GPUs from High-level Specifications. *Procedia Computer Science*, 1(1). ICCS 2010.

[13] H. Markram and et al. Introducing the Human Brain Project. volume 7, pages 39 – 42, 2011. Proceedings of the 2nd European Future Technologies Conference and Exhibition 2011.

[14] H. Mühleisen, M. L. Kersten, and S. Manegold. Capturing the Laws of (Data) Nature. *CIDR*, 2015.

[15] G. Piatetsky-Shapiro and C. Connell. Accurate Estimation of the Number of Tuples Satisfying a Condition. In *SIGMOD '84*.

[16] F. Schürmann and et al. Rebasing I/O for Scientific Computing: Leveraging Storage Class Memory in an IBM BlueGene/Q Supercomputer. In *Supercomputing*, Lecture Notes in Computer Science. 2014.

[17] H. Stephan. The World as a Process: Simulations in the Natural and Social Sciences, 2005.