High Performance Computing for City-Scale Modelling and Simulations

Kenichi Soga[1], Gerard Casey[2], Krishna Kumar[3] and Bingyu Zhao[3]

1. University of California, Berkeley, USA
2. University of Cambridge and Arup, UK
3. University of Cambridge, UK

Keywords - Information technology, Infrastructure planning, Mathematical modelling

City-scale Models and Simulations

The 21st Century is witnessing a rapid rise of urbanization both in the developed and the developing world. Cities increasingly need to be able to do more with less in order to provide for the well-being of their citizens in a sustainable way. The promise of Smart City is an emerging ability to understand, to respond to, and to shape human activity at urban population and geographic scales so that a more agile, adaptive, and sustainable urban environment can be created (see Batty et al., 2012; Caragliu et al., 2011; Chourabi et al., 2012; Su et al., 2011 for early adoption of Smart City). To be effective, this requires the predictive power of data-driven modelling and city-scale computational simulations. Recently city-scale simulations are becoming possible thanks to a surge of development in the high-performance computing (HPC) domain including advanced hardware, computational and algorithmic techniques such as domain decomposition across multi-GPUs and multigrid techniques. Advanced high performance computing systems (a billion billion calculations per second) are now becoming available to performance city-scale simulations with micro-scale models of an individual objective (structure, people, vehicle, etc.) (e.g. Sánchez-Medina et al., 2010; Hori, 2011; Zia et al., 2012; Wijerathne et al., 2013; Pijanowski et al., 2014; Yoshimura et al., 2016; Johansen et al., 2017; Lu and Guan., 2017)

Agent Based Modelling is perhaps the most promising modelling paradigm for attempting to understand complex city systems and facilitating a range of different scenario testing exercises (Casey et al., 2017). Figure 1 shows the visualization of our graph-parallel distributed computing tool that simulates city-scale infrastructure networks with hundred-thousand links and millions of agents traversing close to real-time. The tool performs data-analytics on the results from ABM simulations and monitoring data. Transport services are modelled as edges, and the interaction points (e.g. bus stops) are modelled as vertices in a graph network. Real-time and historic large data sets are used to initialize and update the behaviour of infrastructure and the agents. The purpose of the ABM tool is to capture the complex city-scale response from individual agent behaviours. Macro-scale events such as earthquakes influence the weights of the edges on a graph network (e.g. reduced road capacity/road closures update the weights of the edges/ removal of an edge), which in turn affects the behaviour of individual agents changing the

response of a city. The real-time sampling feature allows for querying and updating individual agent behaviours in a scenario-testing of global impacts, such as to understand the resilience of a city to an earthquake scenario.

Figure 2a shows a city-scale model of London with 300,000 nodes and traffic levels at 8AM on a typical weekday. Five million agents are simulated at a time resolution of 1 hour to traverse the city using various origin-destination pairs using HPC. A road capacity can be decreased or a road can be removed to model infrastructure failure, as shown in Figure 2b.

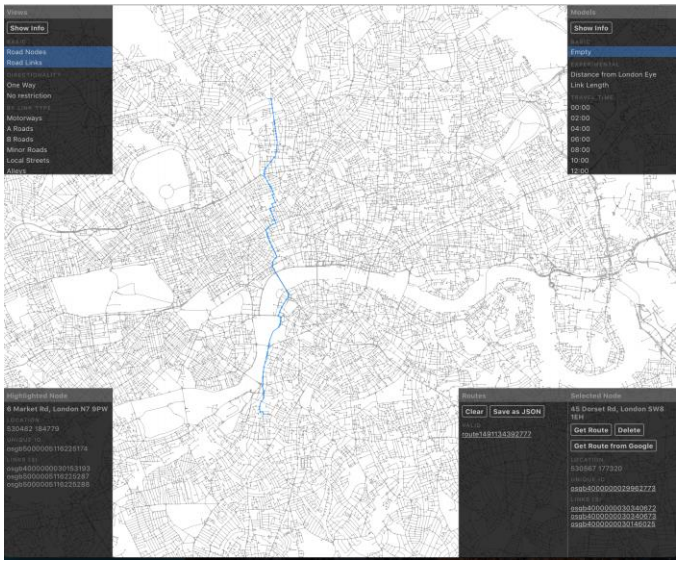
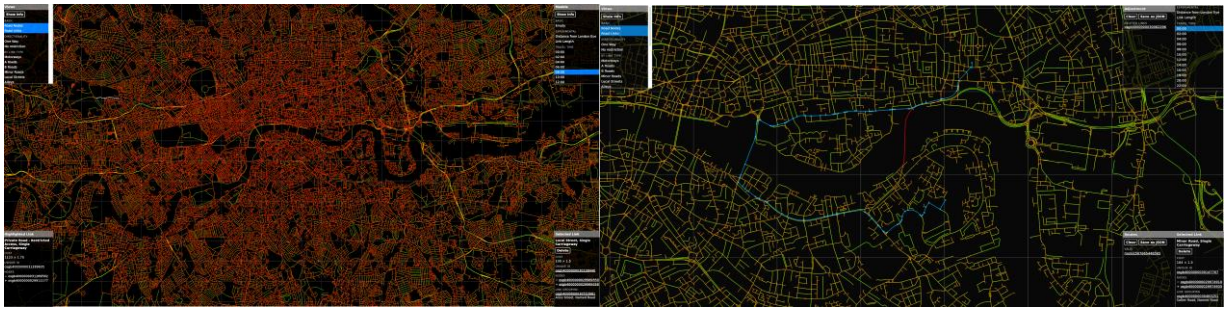


Figure 1 An example shortest path query for travel via car



(a) (b)

Figure 2 City-scale ABM tool - (a) Browser-based visualization of journey times on a typical working day at 8AM in London, (b) Micro-impact assessment of a bridge closure at mid-night in London

Another example of city-scale computing tool is the high-resolution finite element modelling of ground motion to analyse the response of structures during an earthquake. Yoshimura et al.

(2016) performed seismic wave amplification simulation for an urban area of 2.0 x 2.0 km in central Tokyo; this region consists of businesses, shops, and densely built residential districts, with a total of 13,275 buildings. The ground structure has a rectangular shape of 2.0 x 2.0 km and a depth of 100 m. The ground structure is modelled with soil layers using the 5-m grid elevation map and digital geotechnical database. Figures 3a, c and d show the response of ground at the surface as well as the response of the modelled structures. Fig. 3b shows that ground motion responses are far from being uniform and resulting structures shaking change. Such a complexity in the coupled seismic-structure response distribution in an urban area can only be realized by combining high-resolution 3-D ground motion analysis and structure response analysis using HPC.

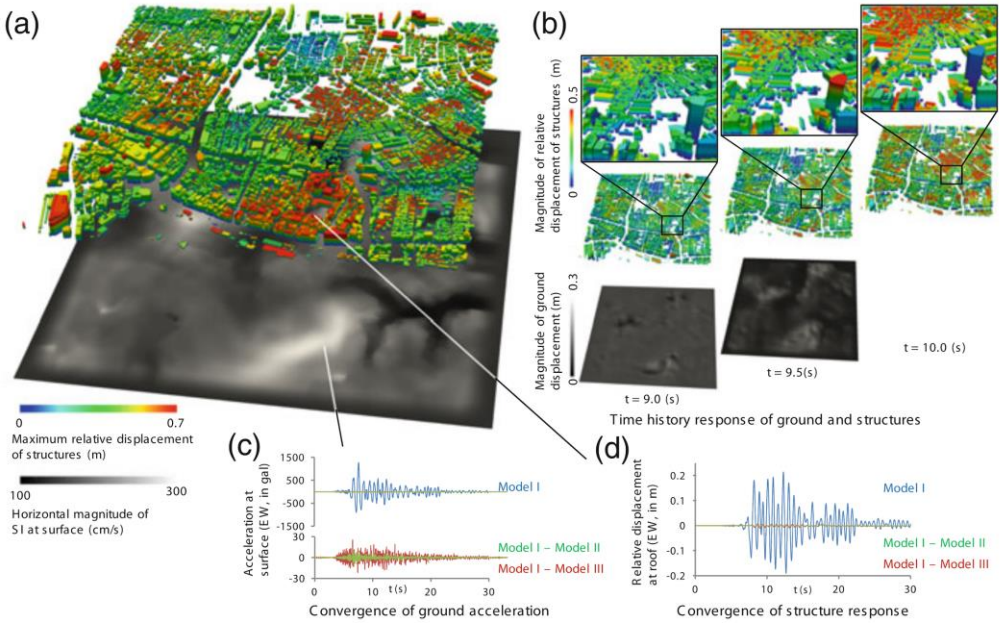


Figure 3 Response of ground and structures of central Tokyo (a) Horizontal magnitude of SI at surface and maximum relative displacement of structures. (b) Snapshots of time history response. (c) Acceleration waveforms at surface (Point A) obtained from Model I (10.7 Billion DOF model), Model II (with element sizes 1.5 times larger than Model I), and Model III (with element sizes 3 times larger than Model I). (d) Relative displacement waveforms of structure A (a two-story RC building located above Point A) obtained using acceleration waveforms computed using Models I, II, and III).

High Performance Computing (HPC)

For city-scale modelling of interactions among people, infrastructure and environment in natural or manmade disasters using HPC, each component of a city has different characteristics and computational requirements (in terms of spatial and temporal resolution, spatial dimension, computational tool, etc.) (Takeuchi et al., 2003; Kitano et al., 1999).

The computation performance of an HPC program depends on the hardware, the modelling paradigm and the software implementation. Until a decade ago, the speed of processors, following Moore's observation, has doubled approximately every 18 months, when the technology hit the energy consumption and heat dissipation wall (Waldrop, 2016). The HPC community moved towards multi-core architecture and was revolutionized by the appearance of General Purpose Graphics Processing Units (GPGPUs). Several large supercomputers now use GPUs as accelerators. For example, the Cray TITAN supercomputer at Oak Ridge National Laboratory, the US's fastest supercomputer in TOP500 - November 2016, has been accelerated by NVIDIA Kepler GPUs. Latest NVIDIA Pascal GPUs have bidirectional interconnect accelerating the bandwidth. In 2012, Intel announced the Many Integrated Core (MIC) architecture product - Xeon Phi coprocessor, which is highly parallel and can be used to accelerate general purpose computing. Xeon Phi uses a Symmetric Multiprocessor (SMP) on a chip that has cache coherence and Uniform Memory Access (UMA). NERSC Cori (Cray XC40), ranked 5th in the Top500, is accelerated using Intel Xeon Phi coprocessors (https://www.top500.org/).

Over the last 25 years, Message Passing Interface (MPI), a HPC standard to efficiently communicate layout of data in memory over the network, has supported a substantial majority of supercomputing work. With the advent of MIC and GPGPUs, there is need to improve the performance on a single node and to efficiently utilize all available compute cores. Shared Memory Multiprocessing paradigms such as OpenMP, OpenACC, Intel TBB, etc have gained traction in the world of traditional HPC. As compilers become efficient in identifying vectorizable regions of codes, there is an increased focus in compiler auto-parallelization. Parallel programming frameworks such as KOKKOS and Charm++ focus on improving performance portability across multiple shared and distributed memory architectures (Edwards et al., 2014; Kale 1993).

As the complexity of research tools increases, the development environment and dependencies are critical for reproducibility. In the past few years, there has been an increased adoption of container and orchestration technologies such as Docker, Singularity and Kubernetes. Containers wrap up a piece of software in a complete file system that contains everything it needs to run: code, runtime, system tools, system libraries – anything you can install on a server. This guarantees that a container will always run the same, regardless of the environment it is deployed (Berstein, 2014). Singularity (http://singularity.lbl.gov/) is an implementation of a container and is an engine to run those containers without requiring any privileged access (root-access). Containers allow researchers to isolate the software environment needed to produce a result away from the configuration and operating system of the computer that the analysis will be running on. As the compute jobs become more complex, resource management and scheduling tools such as YARN and Mesos becomes increasingly relevant.

Data Management

Internet-scale data results in high computational demands. Solving a *Big Data* problem in HPC is challenging. In HPC, large and homogenous arrays are read from a parallel disk and processed with symmetric resources. On the other hand, big data are often heterogeneous and unstructured data, often available in real-time, located in a distributed setting and processed with asymmetric resources. In the mid-2000s, analysing and interpreting real-time streaming data became mainstream along with scalability, large-scale data storage, distributed computing, and fault tolerance (Hashem et al., 2015; Chen et al., 2014). Traditional MPI-based applications had the wrong level of abstraction for I/O intensive/fault-tolerant applications. Message-passing based frameworks soon became very important platforms for building complex parallel and distributed software in different communities. For example, Erlang (https://www.erlang.org/), released to the public just five years later, is a functional language with message-passing built in that has played a very large role in many communications and control environments. Akka (https://akka.io) is a Scala-based message passing framework which supports distributed data model and cluster shrading for processing transaction processing of Big Data. Apache Spark (https://spark.apache.org/) is a general purpose cluster engine for big data processing with a resilient distributed dataset (RDD) capable of supporting graph frameworks. Spark optimizes the tasks running on each file according to its size to fully utilize the cluster. Fundamentally the RDDs are stored in partitions and are operated in parallel using cached data in-memory, which allows for multiple operations across the same data. Spark is often used alongside Hadoop's data storage module (HDFS). Hadoop as a big data processing technology has been around since 2011 and has proven to be the solution of choice for processing large data sets. Hadoop offers distributed fault-tolerant and persistent distributed file system and data recovery capabilities (Hashem et al., 2015).

The emergence of big data has led to a shift in how data is stored, processed and analyzed. Modern graph processing systems apply a vertex-centric logic to transform data on a graph and exploit the graph structure to achieve more efficient distributed execution (Malewicz et al., 2010). Graphs are ubiquitous; the volume and diversity of huge graphs with billions of entities and relationships are a driving force for the development of powerful and highly parallel big data systems such as GraphX (https://spark.apache.org/graphx/). Graph database systems consider graphs as first-class citizens in their data model, query languages, and APIs. They also provide the ability to insert, delete and modify data in a graph database using transactional semantics in addition to running global algorithms on the network such as shortest path algorithms, as well as implement appropriate integrity constraints over the graph (Angles, 2012; Vicknair et al., 2010; Angles and Gutierrez, 2008). However, graph algorithms such as shortest path queries remain mathematically hard problems. GPU and algorithmic implementations show the most promise to solve these problems.

Traditional SQL database pose scalability issues when querying massive geo-spatial datasets. In order to parallelize data processing, a lightweight data interchange format, JavaScript Object Notation (JSON) can be used to distribute data across the compute nodes (Cattell, 2010). The data processing when done in memory is not restricted by constant reading and writing of results to a database. This enables a simulation of large-scale complexity to be possible in a reasonable time frame. Elasticsearch (https://www.elastic.co/) is a distributed search engine capable of performing near real-time searches, by using shared indices with replicas distributed across several compute nodes. Data serialization frameworks such as Apache Avro (https://avro.apache.org) improves the efficiency of data processing (Maeda, 2012). While frameworks such as Apache Cassandra (http://cassandra.apache.org) a highly-available, linearly scalable data store, which is optimized for write intensive workloads (Chebotko, 2015).

City-scale simulations require storing large-quantities of a diverse data set. HDF5 (https://support.hdfgroup.org/HDF5/) is an I/O middleware library and file format for storing and managing data. HDF5 supports an unlimited variety of datatypes and is designed for flexible and efficient I/O and for high volume and complex data (Folk et al, 2011). MPI implementations are good for optimizing various access patterns in applications, and the MPI standard supports complex data types for scientific data. However, MPI does not store any "metadata" for applications, such as the datatypes themselves, the dimensionality of an array, names for specific objects, etc. HDF5 therefore offers the ability to handle data at a higher level, thus allowing parallel I/O methods. With Remote Procedure Call, which allows systems to request services located on other machines through the network, HDF5 calls on one machine can be executed on another. Tiered storage architectures with "Burst Buffers" are gaining more traction, which will be the norm for systems. HDF5 will need to undergo major changes to the data model and file format to fully leverage such systems. Research is ongoing incorporating concepts such as data movements, transactional I/O, indexing, and asynchrony to HDF5 for HPC.

Visualization

Real-time visualization of scientific city-scale simulations is a necessity. I/O in HPC application has long been a bottleneck. Accessing, extracting and assessing the scientific content of large-scale simulation data sets, at terabytes scale, requires specialized tools and techniques. It is not always meaningful to post-process a simplified version of the data. In-situ visualization libraries such as ICARUS and ParaView Catalyst (https://www.paraview.org/in-situ/) examine and query regions of interest and also render simulations in real-time. A loosely coupled and push-driven approach is adopted to link simulation and visualization using a shared memory mapped file as an interconnect. This technology allows for transparent data communication across the network using a client (simulation) / server (postprocessing) model using sockets or MPI layers, thus enabling the ability to pause, examine, slice and visualize real-time simulations (Rivi et al., 2012).

Visualization of complex networks with spatio-temporal information always poses a complex challenge. A web-based interface using WebGL and JavaScript API provides cross-platform functionality. WebGL (https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API) is a cross-platform, low-level JavaScript API based on OpenGL for rendering 2D and 3D graphics in any compatible browsers without the need for plug-ins. CityGML is an XML based open standardized data model and exchange format to store digital 3D models of cities and landscapes. CityDB (https://www.3dcitydb.org) is an open source 3D city database built on CityGML for 3D city visualization on the web using a JavaScript API. Frameworks such as Open Street Maps (https://www.openstreetmap.org) and LeafletJS (http://leafletjs.com/) provide the ability to display data sources and simulations results on city scale models using an Open Standard. Advanced ray-tracing algorithms, such as those used in Disney's Hyperion rendering engine, are able to simulate and visualize cities without the need to downsample or remesh the city (Eisenacher et al., 2013).

Challenges

There are two challenges for large scale simulations using HPC: (i) scalable algorithms and (ii) optimization of I/O through data parallelism.

Algorithms are no longer merely required to communicate across compute nodes, but to efficiently couple with Shared Memory resources within each node. Sustainable future-proof compute algorithms have to be performance portable that can efficiently run on various current and future memory caching layouts and compute architectures.

Data parallelism is almost always I/O intensive (more than 50% and in some cases 98%). The amount of data to visualize is typically of the order of total memory. There are two main factors in visualizing large data: (i) how much data that needs to be read and (ii) how fast can it be read. Relative I/O, which is the ratio of the total memory to the I/O, is quickly becoming a dominant cost in the supercomputing application. In-situ data processing, coupling visualization and analysis routines with the simulation code (no I/O), will enable large data visualization. A tightly coupled system of synchronous co-processing is highly constrained by the memory and impacts performance and prone to crashes. However, a loosely coupled system requires knowing the regions of interest a priori. A hybrid approach means that the data is reduced to a tightly coupled setting (visualization and analysis have direct access to the memory of simulation code) and sent to asynchronous concurrent resources. The Scalable Data Management, Analysis and Visualization institute at Lawrence Berkeley National Laboratory is developing in-situ processing and parallel-IO frameworks to enable software based visualisation on large-scale computing facilities.

A multi-layered graph could be used to model different levels of the system, such as the infrastructure networks and the relationship between these different layers using real-time and

historic large data sets. For example, the London transport graph consists of 423,541 links at the atomic level. There are various sources of data for city-scale simulations. Dynamic streaming data have to be processed in real-time, examples include a dynamically computed General Transit Feed Specification considering real-time feeds at a rate of 24 GB / day, public transit data sourced at 30 second intervals from Transport for London (TfL) and vehicular traffic data is sourced at 2 hour intervals with 200,000 API requests at a rate of 1 GB / day. Other data inputs such as survey, census information, routing schedules are passive sources and are typically batch processed. Crowd-sourced data, which includes social network and transport data, are available at such scale and the efficient use of these data sources to analyse and predict the behaviour of smart cities require asynchronous, concurrent processing on large-scale HPC systems.

Closure

The idea that more information in more dynamic and iterative ways gives better prediction of infrastructure status, transportation behaviour and people movement is central to the promise of the progression of city-scale models using high performance computing. It is hypothesized that, via new data sources such as remote sensors and mobile phones, the reliance on heavily simplified generalised functions for model inputs will be erased. This trade-off between idealised and actual empirical data will be matched with dynamic models that consider complexity at a fundamental level, inherently mirroring the systems they are attempting to replicate. High performance computing (HPC) brings the possibility of doing all of this in less time than the simplified crude models of the past. The hope is to have better answers at the time of important decision-making junctures. Whether this hypothesis is true or not to realise the concept of smart infrastructure and city remains to be seen but it is worthwhile pursuing this initiative.

References

Angles, R. and Gutierrez, C., 2008. Survey of graph database models. *ACM Computing Surveys (CSUR)*, *40*(1), p.1.

Angles, R., 2012, April. A comparison of current graph database models. In *Data Engineering Workshops (ICDEW), 2012 IEEE 28th International Conference on* (pp. 171-177). IEEE.

Batty, M., Axhausen, K.W., Giannotti, F., Pozdnoukhov, A., Bazzani, A., Wachowicz, M., Ouzounis, G. and Portugali, Y., 2012. Smart cities of the future. *The European Physical Journal Special Topics*, *214*(1), pp.481-518.

Bernstein, D., 2014. Containers and cloud: From lxc to docker to kubernetes. *IEEE Cloud Computing*, *1*(3), pp.81-84.

Caragliu, A., Del Bo, C. and Nijkamp, P., 2011. Smart cities in Europe. *Journal of urban technology*, *18*(2), pp.65-82.

Casey, G., Soga, K., Silva, E., Guthrie, P. and Kumar, K., 2017. *A Scalable Agent Based Multi-modal Modeling Framework Using Real-Time Big-Data Sources for Cities* (No. 17-05941).

Cattell, R., 2011. Scalable SQL and NoSQL data stores. *Acm Sigmod Record*, *39*(4), pp.12-27.

Chebotko, A., Kashlev, A. and Lu, S., 2015, June. A big data modeling methodology for Apache Cassandra. In *Big Data (BigData Congress), 2015 IEEE International Congress on* (pp. 238-245). IEEE.

Chen, M., Mao, S. and Liu, Y., 2014. Big data: A survey. *Mobile Networks and Applications*, *19*(2), pp.171-209.

Chourabi, H., Nam, T., Walker, S., Gil-Garcia, J.R., Mellouli, S., Nahon, K., Pardo, T.A. and Scholl, H.J., 2012, January. Understanding smart cities: An integrative framework. In *System Science (HICSS), 2012 45th Hawaii International Conference on* (pp. 2289-2297). IEEE.

Edwards, H.C., Trott, C.R. and Sunderland, D., 2014. Kokkos: Enabling manycore performance portability through polymorphic memory access patterns. *Journal of Parallel and Distributed Computing*, *74*(12), pp.3202-3216.

Eisenacher, C., Nichols, G., Selle, A. and Burley, B., 2013, July. Sorted deferred shading for production path tracing. In *Computer Graphics Forum* (Vol. 32, No. 4, pp. 125-132). Blackwell Publishing Ltd.

Folk, M., Heber, G., Koziol, Q., Pourmal, E. and Robinson, D., 2011, March. An overview of the HDF5 technology suite and its applications. In *Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases* (pp. 36-47). ACM.

Hashem, I.A.T., Yaqoob, I., Anuar, N.B., Mokhtar, S., Gani, A. and Khan, S.U., 2015. The rise of

"big data" on cloud computing: Review and open research issues. *Information Systems*, *47*, pp.98-115.

Hori, M. 2011. Introduction to Computational Earthquake Engineering (2nd Edition), Imperial College Press, 440pp.

Johansen, H., Rodgers, A., Petersson, N.A., McCallen, D., Sjogreen, B. and Miah, M., 2017. Toward Exascale Earthquake Ground Motion Simulations for Near-Fault Engineering Analysis. Computing in Science & Engineering, 19(5), pp.27-37.

Kale, L.V. and Krishnan, S., 1993, October. CHARM++: a portable concurrent object oriented system based on C++. In *ACM Sigplan Notices* (Vol. 28, No. 10, pp. 91-108). ACM.

Kitano, H., Tadokoro, S., Noda, I., Matsubara, H., Takahashi, T., Shinjou, A. and Shimada, S., 1999. Robocup rescue: Search and rescue in large-scale disasters as a domain for autonomous agents research. In *Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on* (Vol. 6, pp. 739-743). IEEE.

Kitchin, R., 2014. The real-time city? Big data and smart urbanism. *GeoJournal*, *79*(1), pp.1-14.

Lu, X. and Guan, H. 2017. *Earthquake Disaster Simulation of Civil Infrastructures From Tall Buildings to Urban Areas*, Springer. 440pp.

Maeda, K., 2012, May. Performance evaluation of object serialization libraries in XML, JSON and binary formats. In *Digital Information and Communication Technology and it's Applications (DICTAP), 2012 Second International Conference on* (pp. 177-182). IEEE.

Malewicz, G., Austern, M.H., Bik, A.J., Dehnert, J.C., Horn, I., Leiser, N. and Czajkowski, G., 2010, June. Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data* (pp. 135-146). ACM.

Pijanowski, B.C., Tayyebi, A., Doucette, J., Pekin, B.K., Braun, D. and Plourde, J., 2014. A big data urban growth simulation at a national scale: configuring the GIS and neural network based land transformation model to run in a high performance computing (HPC) environment. *Environmental Modelling & Software*, *51*, pp.250-268.

Rivi, M., Calori, L., Muscianisi, G. and Slavnic, V., 2012. In-situ visualization: State-of-the-art and some use cases. *PRACE White Paper*, pp.1-18.

Sánchez-Medina, Javier J., Manuel J. Galán-Moreno, and Enrique Rubio-Royo. "Traffic signal optimization in "La Almozara" district in Saragossa under congestion conditions, using genetic algorithms, traffic microsimulation, and cluster computing." *IEEE Transactions on Intelligent Transportation Systems* 11, no. 1 (2010): 132-141.

Su, K., Li, J. and Fu, H., 2011, September. Smart city and the applications. In *Electronics, Communications and Control (ICECC), 2011 International Conference on* (pp. 1028-1031).

IEEE

Takeuchi, I., Kakumoto, S. and Goto, Y., 2003, July. Towards an integrated earthquake disaster simulation system. In *First International Workshop on Synthetic Simulation and Robotics to Mitigate Earthquake Disaster*.

Vicknair, C., Macias, M., Zhao, Z., Nan, X., Chen, Y. and Wilkins, D., 2010, April. A comparison of a graph database and a relational database: a data provenance perspective. In *Proceedings of the 48th annual Southeast regional conference* (p. 42). ACM.

Waldrop, M.M., 2016. The chips are down for Moore's law. *Nature*, *530*(7589), pp.144-147.

Wijerathne, M.L.L., Melgar, L.A., Hori, M., Ichimura, T. and Tanaka, S., 2013. HPC enhanced large urban area evacuation simulations with vision based autonomously navigating multi agents. *Procedia Computer Science*, *18*, pp.1515-1524.

Yoshimura, S., Hori, M. and Ohsaki, M. eds., 2016. *High-Performance Computing for Structural Mechanics and Earthquake/Tsunami Engineering*. Springer International Publishing.

Zia, K., Riener, A., Farrahi, K. and Ferscha, A., 2012, July. A new opportunity to urban evacuation analysis: very large scale simulations of social agent systems in Repast HPC. In *Proceedings of the 2012 ACM/IEEE/SCS 26th Workshop on Principles of Advanced and Distributed Simulation* (pp. 233-242). IEEE Computer Society.