Acceleration and visualization of Dynamic Network Optimization

Acceleration and Visualization of Dynamic Network Optimization

Yuanzhou Ye[#φ], Oswaldo Cadenas[#], Graham Megson[*]
[#]School of Systems Engineering, University of Reading / (Arieso Ltd[1]), Reading, United Kingdom
[*]School of Electronics and Computer Science, University of Westminster, London, United Kingdom
[φ]Symantec Corp., 350 Brook Drive, Green Park, Reading, United Kingdom

***Abstract* - With the emerging prevalence of smart phones and 4G LTE networks, the demand for faster-better-cheaper mobile services anytime and anywhere is ever growing. The Dynamic Network Optimization (DNO) concept emerged as a solution that optimally and continuously tunes the network settings, in response to varying network conditions and subscriber needs. Yet, the DNO realization is still at infancy, largely hindered by the bottleneck of the lengthy optimization runtime. This paper presents the design and prototype of a novel cloud based parallel solution that further enhances the scalability of our prior work on various parallel solutions that accelerate network optimization algorithms. The solution aims to satisfy the high performance required by DNO, preliminarily on a sub-hourly basis. The paper subsequently visualizes a design and a full cycle of a DNO system. A set of potential solutions to large network and real-time DNO are also proposed. Overall, this work creates a breakthrough towards the realization of DNO.**

Keywords: Dynamic Network Optimization, Self-Organizing Network, Self-Optimization, Mobile Network Optimization Algorithms, Cloud Parallelization, Inter Process Communication

I. INTRODUCTION

The past 5 years have witnessed an emerging prevalence of smart phones, tablets and 4G LTE networks. Mobile network operators can no longer afford to be outpaced by the varying demands presented by subscribers. The tightened budgets as well as flattened mobile penetration rates make it essential for service providers to operate their networks as efficiently as possible at all times. They have been pursuing aggressive tools for automated Mobile Network Optimization (MNO) that optimally determines the network settings such as base station transmit power and azimuth that best serve the demand within budgetary constraints [1]. Nevertheless, MNO is classed as a Non-deterministic Polynomial-time (NP)-hard combinatorial optimization problem [2] that is highly compute and data intensive. Consequently, timescales associated with MNO are commonly in tens of hours, with the ever-growing network complexity. MNO is largely static, yet network traffic exhibits significant spatial and temporal variations, for instance, the commute-work-home routines of most subscribers. This fundamental limitation has inspired the Dynamic Network Optimization (DNO) concept to emerge that aims to continuously and optimally configure the network settings to adapt to changing network conditions and demands [3][4].

The DNO development, however, is still at infancy, largely hindered by the lengthy optimization runtime. Likewise, the key functionality "self-optimization" of the Self-Organizing Network (SON) technology [5] faces the same bottleneck. Our long-term goal is to realize DNO on a sub-hourly basis, implying that the optimization phase of a DNO cycle ought to complete ideally within a quarter of an hour regardless of the network size and complexity. Our prior work [6][7] has designed and implemented scalable distributed parallel solutions for two core MNO algorithms that achieved substantial speed gain at 7.5 and 14.5 on a 16-core distributed system. GEOson (originally known as ariesoACP) from Arieso Ltd [8], the market-leading automated MNO product, serves as the testbed for acceleration in our work. Yet, given a large and loaded network such as central London, acceleration of optimization from a scale of tens of hours to sub-hour is challenging and would require an equally massive distributed system, presumably upon hundreds of thousands of processors. For network operators, such a system is neither cost-effective nor easily manageable. The first part of this paper further addresses the scalability of the distributed system and enhances the efficiency and robustness of the Inter-Process Communication mechanisms. The second part visualizes a preliminary design and a full cycle of a DNO system alongside various approaches to realize it.

II. METHODOLOGIES AND TECHNOLOGIES

A. Acceleration of Mobile Network Optimization Algorithms

As an NP-hard multiple-objective constrained combinatorial optimization problem, MNO problems exhibit high complexity that requires applying iterative heuristic algorithms. Our prior work has managed to accelerate two core MNO algorithms, Intelligent MNO Algorithm [6] that is an iterative heuristic algorithm with intelligent learning capabilities and greedy MNO algorithm [7] that only begins upon the completion of the "Intelligent" algorithm. We proposed and implemented novel predictive binary or greedy tree patterns to expose the parallelism. The parallel designs

1 Project partly funded by Knowledge Transfer Partnership (KTP), between the University of Reading and Arieso Ltd

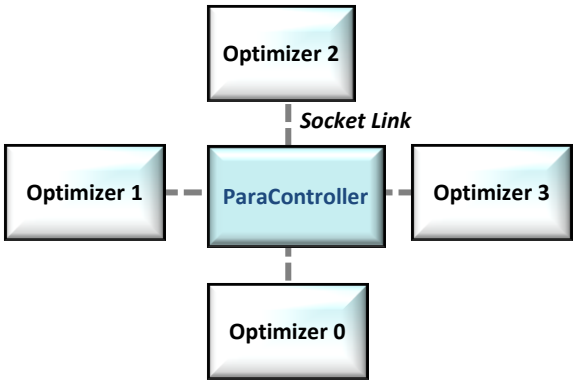

Fig. 1. Diagram of a Simple Instance of the Distributed Parallel System

are deployed on a distributed system that adopts the server-client model utilizing the computer cluster. Fig. 1 presents a simple instance of such a distributed system. The parallel controller along with a shared folder is hosted by the server node whilst optimizers reside on the client nodes. The IPC mechanism is via Windows sockets, and PsExec utility serves as the main technique for remote process control. The cluster based system manifests good scalability, but it becomes costly and hardly deployable in the context of tens of thousands of nodes for DNO. A desirable system would be one that can scale on demand that is to scale up to its full potential for large networks and scale down to fit small and medium ones. Thanks to high scalability and elasticity, Software as a service (SaaS) [9] based cloud platform stands out as a perfect match.

B. Cloud Parallelization

A cloud based parallel solution exhibits considerable beneficial characteristics. From the end users' viewpoint, MNO tools can be delivered as a service and accessible via a thin client interface such as a web browser through the internet, which is much simplified as opposed to the physical software based solution. The dynamic nature of cloud bridges the gap between the relatively static optimization process and the dynamics needed for DNO. By means of a unified web application, one can establish a full cycle of DNO that network operators feed traffic and network data into the cloud and the MNO service performs optimization jobs behind the scenes and suggests optimal network configurations post the optimization. Further, deployment and initialization of MNO tools can be simplified. At present, users have to deploy the optimization software on each node of the distributed system and a ParaController process remotely controls the launch of all optimizers. These procedures produce high overhead, and hence it is beneficial to redesign and pre-install the MNO processes as a service. Moreover, the cloud solution possesses fault tolerance and high availability as a faulty node can be easily replaced by a failover node in the cloud such that it allows speedy recovery.

However, we are facing tremendous challenges in terms of the migration of our MNO applications into the cloud. The first is how to implement MNO processes as a service whereby it will manage the start and stop of optimizer processes. The second is how to improve the IPC mechanism since Windows sockets may not be as scalable and reliable and how to address the limitations placed by the single-controller system when the system size grows. The third is how to support parallel file access as shared folders and files are being used to retain certain network data for replication but may not be efficient for a large system.

III. Design and Prototype of the Cloud Parallel Solution

This section illustrates the design and prototype of the cloud parallel solution. Since the MNO algorithms were written in C++, the software development language remains C++.

A. System Deployment and Initialization

To migrate our MNO applications into the cloud, a first step is to simplify the early deployment and initialization. A new MNO process is created as a Windows service that is effectively a shell process. The optimizer binary is still required to be pre-deployed in a fixed folder location on each compute node as before. However, this deployment only needs to occur once and the main simplification is that the optimizer executable no longer needs to be launched remotely. The new MNO service is configured to automatically run when the operating system starts, and it invokes a pool of local optimizer processes whose number defaults to the number of cores on the cloud node. By using a local process pool, the overhead for remote process control is eliminated and there is no need for users to specify the remote location of the optimizer binaries, and the ParaController process is now freed to mainly handle the synchronization among optimizers. The optimizer process is redesigned to wait while consuming little computing resources such as CPU until optimization jobs have been requested by a user.

B.Multi-ParaController Design

In the world of cloud, as the system size is constantly growing, IPC plays a key role to high performance. A closer look at the single-controller design by Fig.1 reveals inefficiency when talking to an enormous amount of optimizers across the network. Every IPC call between the ParaController and optimizers is a remote call across machine boundaries and presents higher cost. A solution is to introduce multiple ParaControllers whilst maintaining a Main-ParaController. Initially a viable approach is to have a ParaController per compute node since the communication becomes solely local and ParaController itself is a lightweight process. At a later stage, we may consider an enhanced solution as to have one controller per optimizer compute group that comprises closely positioned nodes. Diagram of a basic multi-controller based cloud parallel system is given by Fig. 2. Research has been carried out in search of a more efficient and secure IPC mechanism to replace sockets and Component Object Model (COM) and Distributed COM (DCOM) technologies are chosen, also as shown in Fig. 2.

C. Redesign of Inter-Process Communications Mechanism

COM is a standard for software component reutilization [10].

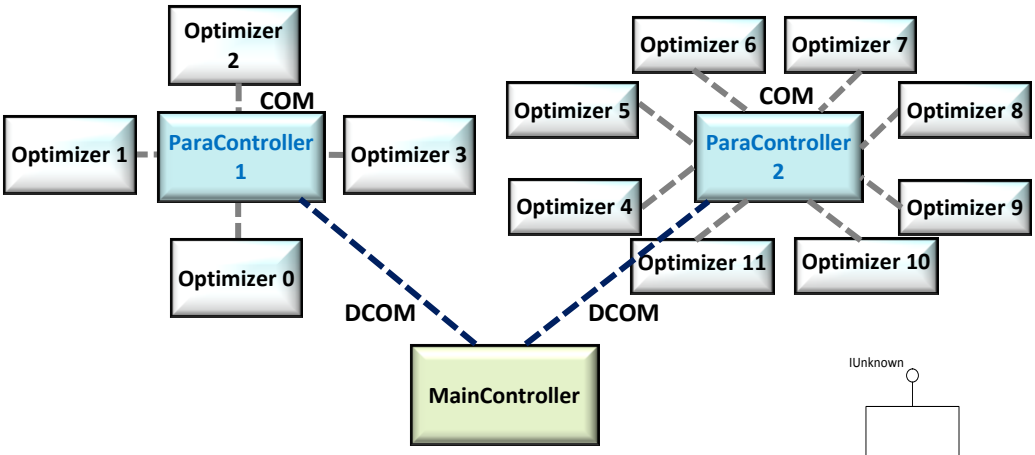

Fig. 2. Diagram of a Multi-Controller based Cloud Parallel System

As our MNO algorithms are implemented in C++, COM and DCOM are chosen given its better compatibility and supportability through .NET COM Interop if we migrate to .NET later. An important characteristic of COM is its location transparency to applications that the client and server can establish IPC irrespective of their relative locations.

ParaController, Optimizer and the new MainController executables are re-implemented as COM EXE servers whose COM diagrams are in Fig. 3. Note that IUnknown is an interface that all COM components must implement. The ParaController COM component implements IParallelControl interface that includes the following methods:

HRESULT Init([in] networkConfigXML, [in] paraControllerID): is called by MainController and, as part of the initialization, a cache of ParaController COM objects are created.

HRESULT NotifySolutionAccepted([out] costData): is called twice by an optimizer to accept a solution, one to its owning ParaController and in turn another to the MainController.

HRESULT CheckSolutionAccepted([out] hasSolutionAccepted, [out] costData): requests all ParaControllers to notify related optimizers to check whether a solution of current iteration is accepted. A pre-selection of the best solution is performed locally to a ParaController and results are returned to the MainController to choose the overall best one for replication.

HRESULT Stop(): stops the optimization of all optimizers.

HRESULT Get_Result([out] optResultXML): returns optimization result (network configurations, Key Performance Indicators (KPIs) and such) in XML format to the cloud client.

The IMainControl interface contains a set of wrapper methods for Stop, Init and NotifySolutionAccepted with almost identical signatures as on IParallelControl. The optimizer COM component implements a set of interfaces, namely IStop and IParallelComms whose functionalities basically match the request-reply pairs in the original IPC mechanism of our distributed solution [6].

HRESULT InitOptimization([in] networkConfigXML, [in] optimizerID): assigns a unique optimizerID to each optimizer who initializes the network and kicks off optimization.

HRESULT CheckSolutionAccepted([out] hasSolutionAccepted, [out] costData): requests all optimizers, to check whether a solution is accepted. If the answer is yes, the corresponding optimizer sends costData values back to ParaController.

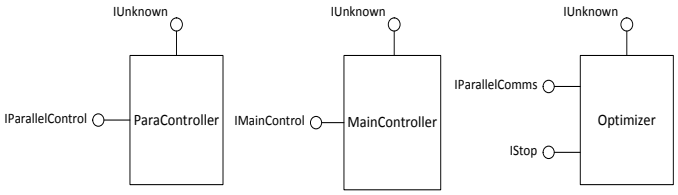

Fig. 3. COM Diagram of Controller and Optimizer Components (coclasses)

HRESULT WriteReplicationFile(): is called by the MainController to the selected ParaController that in turn calls the best optimizer.

HRESULT ReadReplicationFile(): notifies non-best ParaControllers to request their managed optimizers to perform replication of the best network configurations.

At a glance, it appears that more communications are involved but in essence remote calls are significantly reduced and the system becomes more manageable and hence robust.

D. Parallel File Access Enhancement

Our cloud system uses the Network File System (NFS) to store shared data that encompasses a file for replication of network configurations and an optimization result file. These files do not support concurrent writing and reading access that could become a bottleneck for scalability of cloud and High-Performance Computing (HPC) cloud platforms. Also, a faulty file server node could easily cause data loss and system paralysation. An option is to introduce multiple copies of the files over the network, in place of a centralized storage location. In such a manner, optimizers can read from or write to files stored on nodes that are physically closer, and data loss can be mitigated at a light cost of data duplication. Extensive research discovers that Parallel File System (PFS) techniques such as IBM General PFS (GPFS) [12] could be employed to our design. Distributed PFS stripes chunks of data from an individual file over multiple file server nodes such that different data chunks can be read and written in parallel. Hence, our cloud system can output optimization outcomes from the result files frequently to the client interface, without blocking optimizers from writing the latest statistics into the same result files. This results in higher data throughput.

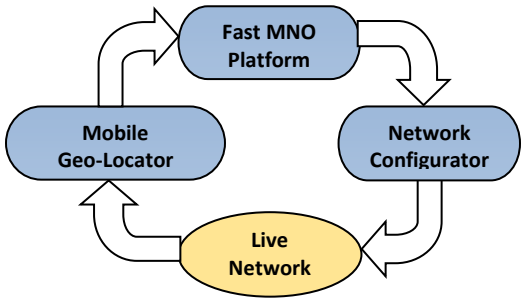

Fig. 4. A Basic Design of a DNO system

IV. Visualization of Dynamic Network Optimization

The demand for high data rates has driven the infrastructure of certain 3G and emerging 4G LTE networks toward micro-cellular structures. This introduces intrinsically greater traffic fluctuations than traditional voice traffic. DNO can address the dynamic demands as well as specific events such as a big football match or a traffic jam where a significant spike in mobile usages will be seen in a concentrated area. We hereby visualize a holistic DNO system that comprises three essential components, namely mobile geo-locator, fast MNO platform and network configurator, as depicted in Fig 4. Our research thus far has been focusing upon the design and realization of a fast parallel MNO platform. We will next explain the roles of the other two DNO components.

A. Mobile Geo-Locator

The main purpose of mobile geo-locator is to locate real-time mobile traffic and thereby identify where the problem cells are. Its core technique is based upon Per Call Measurement Data (PCMD) [13]. With such location intelligence, our fast MNO platform can score each cell on its performance by calculating the KPIs or on its voice or data traffic load. Subsequently, optimization starts with the underperformed cells or cells with heaviest load, thereby intelligently accelerating the optimization process to reach KPI targets of the overall network. AriesoGEO platform developed by Arieso Ltd [11] has a proprietary algorithm that reads in raw PCMD records, parses and processes them and outputs the best estimated locations. The output location file is in a format that is recognizable by and can be seamlessly loaded into our research testbed AriesoACP platform.

B. Automatic Network Configurator

The network configurator component is responsible for automatically and remotely adjusting network configurations suggested by our optimization platform. It matches self-optimization functionalities of SON that aim to minimize human interventions and adjust network parameters on a regular basis. This automation becomes viable with the deployment of programmable base station antennas. KWM programmable antenna system is one of the available systems. And antenna management features are being integrated into O&M platforms of 4G LTE networks.

By remotely updating the azimuth, tilt and power output of antennas on a regular basis, the real benefits are that the network coverage topology is dynamically optimized to balance the traffic between congested base stations and underutilized ones, which translates into higher QoS. We will need to standardize an API that can support various vendors.

C. Dynamic Network Optimization on a Sub-Hourly Basis

For small and relatively medium sized networks and specific events such as major traffic congestion or a natural disaster, ideally DNO ought to be performed at a margin of 15 minutes. Given a standard dual-core (at 2.16GHz) computer, the mobile geo-locator can locate 120, 000 PCMD records in approximately a 2-minute span. The figure is customarily sufficient for the number of mobile subscribers simultaneously using various services in central areas of a busy city. The timing consumed by the optimization phase is the determining factor. If we define small and medium sized networks as of less than 30 cell sites, an 11-minute optimization run may be achieved with a set of arrangements:

First, in the course of initialization, KPI calculations are performed per cell using the real-time traffic map from the geo-locator. Each cell is scored and problem cells will be optimized first alongside their neighbouring cells. In such a fashion, the optimization algorithms tend to converge swifter, since the problem cells exhibit more room for optimization.

Second, an asynchronous distributed and multi-threaded hybrid parallel solution based upon our prior work can be applied to boost the speed performance.

Third, the 11-minute is not a concern for event driven DNO as it is performed on a concentrated area where a major traffic jam or sports event resides, whereas for a 30-site geographically sparsely distributed network special arrangements are required. We can decompose the network into small sized ones since radio coverage areas that are far apart have little correlation (e.g. interference) with each other. The partitioned sub-networks can be optimized in parallel and individual results can be aggregated in the end. By partitioning with a carefully selected interference-free distance, a speed gain required by each sub-network is relatively smaller. Consequently, the original problem is transformed into a simpler problem manageable by a few hundreds of nodes.

To guarantee a fixed optimization run-time, a timer can also be set to terminate the optimization precisely after 11 minutes. The remaining 2-minute span will be allocated for the network configurator operations that apply the optimized network configurations to the real network. Remote configurations via electronic means can be accomplished swiftly, yet a fast interface or transport protocol needs to be implemented.

V. Potential Solutions to Large Networks and Real-Time DNO

Due to the enormous data intensity, fast DNO for a large network such as London with over 400 cell sites or real-time DNO may be unviable without a “super” HPC cloud system as of today. Nonetheless, in the long run, there are a collection of approaches that can help realizing real-time DNO:

Again the first is the aforementioned network decomposition techniques that are particularly beneficial to large networks. The scheme to partition a network largely replies upon the shape and terrain of the area under optimization. An interference-free distance can be computed upon the maximal range which a transmitter can interference with a receiver and an empirical study [14] suggests a 2-kilometer distance. The more parallel zones we can partition, the more speed gain we can obtain.

The second is to pre-compute by pattern modelling and store possible solutions in response to various network scenarios or "profiles" into a lookup table or database where the best network configurations could be fetched for a given set of network patterns and KPI targets. The lookup time could be short but a sound algorithm that can find the best match is yet to be designed. In order to adapt to minor changes in the network conditions, an issue could be that the lookup table needs to be populated with a large volume of data, as to store every possible pre-calculated solution. This might be addressable by the big data related technologies.

An alternative to the above approach is to store selected solutions on the fly. In a sense, this approach is network event driven where the notion of network events refers to not only specific events such as a large sports event but also regular events such as traffic variations over various areas throughout a day. For example, once a DNO cycle is performed for a typical event that the mobile traffic shifts from residential areas in the evening into commercial districts during the office hours, its optimized network configurations will be stored and recommended as a good match when a similar event of the network surfaces. The key is its learning capabilities that may become achievable with the assistance of the artificial neural network and machine learning technologies. The difficulty would be in choosing suitable neural network architecture as well as the type of neuron and the training algorithm. Once the network is properly trained and can learn from past DNO runs with diverse network conditions and patterns, it ought to swiftly produce good solutions even for those scenarios that the network has not been trained for.

The last but crucial approach is the adoption of state-of-the-art HPC platforms that can provide sufficient yet affordable parallel computing power. A candidate might be the latest GPGPUs such as Compute Unified Device Architecture (CUDA) by NVIDIA. GPGPUs are much more powerful than conventional CPUs and cost reasonably. Platforms like CUDA can provide tens or hundreds of cores, and hence much more parallel execution pipelines that can concurrently perform tasks of an optimizer per core. The main issue is that many efforts will be required to migrate our C++ implemented optimization algorithms into a varied language and perhaps a different operating system supported by GPGPUs. The overall speed performance is expected to be significantly improved.

VI. Conclusion

The design of a novel cloud parallel solution is proposed and prototyped, and it can scale up in ease to satisfy the high performance requirement of DNO. In comparison with our prior work, the new solution delivers optimization as a service and simplifies the client interface and deployment. The prototype can be tested on a large-scale private cloud platform if our budget allows in the future. To achieve better scalability and reliability than Windows Sockets, a new COM/DCOM based IPC mechanism is introduced and is fully compatible with our C++ backed optimization algorithms. We also present a novel Multi-ParaController design for more efficient IPC in the cloud.

Also proposed is a system design of a full DNO cycle that consists of mobile geo-locator, fast MNO platform and network configurator components. To cater for the surging data demands pressed by the uptake of smart phones and emerging 4G LTE networks, we visualize a realistic sub-hourly DNO cycle in order to dynamically adjust the network to meet changing demands and network conditions. To realize a fixed 15-minute DNO cycle, a number of solutions are suggested: optimization is made more intelligent by starting with "problem" cells and network decomposition schemes are optionally employed depending upon the network size and complexity. To look forward upon large network and real-time DNO, potential solutions are discussed such as pre-computed network solutions, pattern matching using artificial neural network techniques and migration to HPC platform, just to name a few. Overall, this is a milestone towards the realization of DNO as well as fast self-optimization for SON.

References

[1] J. Laiho, A. Wacker and T. Novosad, Radio Network Planning and Optimisation for UMTS, Wiley, 2006, Ch. 8

[2] D. S. Hochba, "Approximation Algorithms for NP-Hard Problems", ACM SIGACT Newsletter, Vol. 28 Issue 2, pp. 40 – 52, Jun. 1997

[3] S.C.Borst, A. Buvaneswari, "Dynamic Optimization in Future Cellular Networks", Bell Labs Tech. J., 10(2), 2005, pp. 99–119

[4] L. M. Drabeck et al., "Network Optimization Trials that Prove a Vendor-Independent Methodology Using Ocelot", Bell Labs Tech. J., 2005

[5] 3GPP, "Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Self-configuring and self-optimizing network use cases and solutions", Technical Report TR 36.331

[6] Y. Ye, G. Megson and O. Cadenas, "Asynchronous Distributed Parallelization of Mobile Network Optimization Algorithms", Proc. Global Wireless Summit, Atlantic City, USA, 2013

[7] Y. Ye, O. Cadenas and G. Megson, "Distributed Parallelization of Greedy Mobile Network Optimization Algorithms", IEEE 21st International Conference on Software, Telecoms and Computer Networks, 2013 (Accepted)

[8] Brief description of GEOson, Arieso Ltd, [Online],
WWW. on http://www.arieso.com/products/applications/geo-son/

[9] W. Chou, "Web Services: Software-as-a-Service (SaaS), Communication, and Beyond", IEEE Congress on Services Part II, 2008

[10] R. Grimes, A. Stockton, Beginning Atl Com Programming, Wrox Press, 1998, Ch. 1

[11] AriesoGEO platform, Arieso Ltd, [Online],
WWW. on http://www.arieso.com/products/platform/

[12] F. Schmuck, R. Haskin, "GPFS: A Shared-Disk File System for Large Computing Clusters", Proc. the FAST'02 Conference on File and Storage Technologies, pp. 231–244, 2002

[13] M. J. Flanagan, L. M. Drabeck, L. A. Cohen, "Wireless network analysis using per call measurement data", Bell Labs Technical Journal, Vol. 11, Issue 4, pp. 307–313, 2007

[14] S. E. Elayoubi, "On the Parallelization of Radio Network Planning Tools", Vehicular Technology Conference, pp. 951-954, 2007