18th International Conference on Space Operations, Montreal, Canada, 26 - 30 May 2025.

SpaceOps-2025, ID # 159

Applying Highly Parallel Computation to Improve Interference Simulation for Mega Constellations

Jordan Richter [a]*, Haley McDermott [b], Jarin Tasnim [c], Tahseen Ahmed [d]

[a] *Calian, Advanced Technologies, Canada*, jordan.richter@calian.com
[b] *Calian, Advanced Technologies, Canada*, haley.mcdermott@calian.com
[c] *Calian, Advanced Technologies, Canada*, jarin.tasnim@calian.com
[d] *Calian, Advanced Technologies, Canada*, tahseen.ahmed@calian.com

* Corresponding Author

Abstract

As we anticipate the proliferation of mega-constellations, organizations tasked with governing frequency usage and planning new satellite missions face a formidable challenge: conducting realistic and comprehensive simulations. These simulations play a pivotal role in assessing potential interference impacts and governing the safe and fair use of spectrum. To provide robust data for mission planning, we sought to develop a highly parallelized tool to handle such complex operations. This project was undertaken with financial support of the Canadian Space Agency. The primary challenge lies in scaling these computations to accommodate future mega-constellations. As the number of satellites increases, the number of calculations not only grows, but the complexity for coordinating resources also grows. Given the sheer volume of computations we recognize that scalability demands a dual approach: both horizontal and vertical scaling. Our simulation is intended to provide three primary sets of functionalities. The first is orbit propagation and contact planning. This stage is responsible for maintaining orbits and ensuring that the best contact opportunities are selected using a scoring system based on the length of a contact. The second stage is beam projection, where vectorized calculations are completed to consider the footprint of satellite beams on areas of interest. Finally, power and interference calculations are completed to gather valuable data on the operation of the system. Our approach leverages high levels of vertical scaling through GPU acceleration and furthers the scalability through horizontal scaling in cluster-based computing. Based on Calian-developed EarthMesh™, the application indexes calculations across an elastic geospatial grid. Through this we achieve significant performance gains compared to classical simulation techniques. While the extent of improvement varies depending on the computational scale and scope, we typically observe speedups ranging from 10 times to over 300 times. Our baseline scenario involves simulating 1000 satellites against 384 ground stations on a single workstation equipped with GPU acceleration constituting a substantial leap from the previous tools that require days to compute much smaller scenarios consisting of less than 20 satellites. Beyond its immediate application of simulating links between satellites and ground stations, our spatially elastic computation engine holds promise for broader use. We are looking towards applying the same performance enhancements to user-focused spaces, where computations span regions or even the entire globe. This versatility could facilitate system performance simulations or pave the way for developing spectrum-sharing technologies across constellations. While ongoing research and refinement remain essential, initial results indicate that this highly parallel solution will be a potent tool for a wide range of satellite communication simulations. As constellations continue to expand in both flexibility and size, our approach promises to keep pace with the demands of our interconnected cosmos.

Acronyms/Abbreviations

CPU – Central Processing Unit
CSA – Canadian Space Agency
ECEF – Earth-centered, Earth-fixed coordinate system
EESS – Earth Exploration Satellite Service
GPU – Graphical Processing Unit
ITU – International Telecommunication Union
LEO – Low Earth Orbit
STDP – Space Technology Development Program
SGP4 – Simplified General Perturbations 4
TLE – Two Line Element
VRAM – Video Random Access Memory

1. Introduction

The radio frequency spectrum is a highly contested and expensive resource. Limited bands of spectrum are allocated for specific use cases and require coordination among multiple users. To govern and predict the safe and fair use of the spectrum, comprehensive and realistic simulated interference data is essential. Generating accurate and thorough simulated interference data necessitates increasingly complex and dynamic spacecraft modeling. Technological advancements, including dynamic beamforming, steerable beams, and fractional frequency re-use, add additional complexity to modeling modern satellite network operations and necessitate powerful simulation tools. Additionally, the number of low Earth orbit (LEO) satellites is rapidly increasing due to surging demand for high-speed, low-latency connectivity and reduced launch costs to orbit [1, 2]. Many LEO satellites belong to planned "mega-constellations" consisting of several thousand satellites. As the space environment becomes increasingly congested with the addition of mega-constellations, the number of calculations required to accurately model interference scenarios trends into the trillions.

The development of the simulation tool was funded by the Canadian Space Agency (CSA) as part of the Space Technology Development Program (STDP). The CSA is responsible for coordinating spectrum management for multiple missions that use the Earth Exploration Satellite Service (EESS) S-band or X-band. The CSA assesses several scenarios over multi-year time frames to ensure that new missions do not exceed acceptable levels of interference with incumbent missions. The CSA utilizes International Telecommunication Union (ITU) standards, such as ITU-R SA.1027-6 [3] and ITU-R SA.609-2 [4], to define and assess allowable levels of interference. To identify worst-case scenarios, sensitivity sweeps over constellation defining orbital parameters are performed, increasing the size of simulations by introducing additional simulation scenario permutations to evaluate.

Computational complexity arises from the number of entities and interactions between those entities. These complex models are expanded in scale by scenario length and permutations. Effective simulation tools must be capable of handling computational complexity and scaling as additional mega-constellations come online. Many existing commercial and open-source simulation tools struggle with efficient performance and timely execution of large-scale complex simulations [5, 6]. Therefore, the problem of modeling satellite networks at scale is well suited for a parallel processing approach. The goal was to create a robust simulation tool that leverages vertical scaling through graphical processing unit (GPU) acceleration and horizontal scaling via cluster-based computing.

This paper describes a parallelized simulation tool developed by Calian that performs orbit propagation, contact planning, beam projection, and power calculations to gather valuable data over a range of scenarios. It highlights the theory, technology, and approach to creating a parallelized, highly scalable simulation tool. Benchmarking results are outlined and discussed, highlighting areas of improved performance. Future areas of investigation are described.

2. Theory and calculation

The simulation tool completes a simulation in four sequential stages:

1. Orbit propagation and contact availability assessment.
2. Contact selection.
3. Power computations.
4. Results analysis.

When all four stages are complete, a simulation report is produced outlining the impacts of interference from the interfering constellation on the victim constellation.

The simulation tool functions on the premise of highly repeatable computations over multidimensional problem sets. These dimensions include geographic space, time, and scenario permutations. Given the size of the problem data set, it is not always possible to represent all dimensions in a single software object. Limited by the scale of resources available within a single compute unit, it was proposed to distribute the computation resources across multiple units. This distribution of computations across both vertical and horizontal parallelism is expected to significantly increase the performance of the computations over time.

2.1 Satellite Network Modelling Ethos

The simulation tool supports the definition of scenarios with individually defined satellites or satellites belonging to constellations. Satellite orbital elements are defined using Keplerian orbital parameters or via Two Line Element (TLE) sets. A selection of several generic orbit propagation models, including simple Keplerian, Eckstein-Hechler,

and Simplified General Perturbations 4 (SGP4), are available and supported through the Orekit library [7]. The simulation tool does not consider orbit-keeping maneuvers over time. Each satellite is assigned an antenna pattern as well as a pointing mode, with options for nadir or tracking pointing.

Ground stations are described with a position, a pointing mode, an antenna pattern, and an operating schedule dictating when the ground station can transmit and/or receive. Periods of operational contact between ground stations and satellites are evaluated and scheduled by the simulation tool. The tool identifies periods when a satellite has line of sight to an operating ground station and exceeds the defined minimum elevation angle. During a simulation, there are often cases where satellites in the same network are in view of the same ground station. The tool uses a scoring system based on the length of contact, favoring longer periods of contact, to deconflict and create a schedule.

Prior to performing power calculations, the simulation tool projects the antenna patterns on areas of interest, considering the satellite and gateway pointing modes. The simulation tool then performs power and interference calculations, including considerations for free space loss and atmospheric loss as defined by ITU-R P.525-4 [8] and ITU-R P.618-10 [9], respectively. Power and interference are calculated as follows:

$$C_0 = EIRP_{TXo} + G_{RX} - L_{FS} - L_{Atmos} \quad (1)$$

$$EIRP_{TXo} = P_{TXo} + G_{TX} \quad (2)$$

$$I_0 = EIRP_{TXo} + G_{RX} - L_{FS} - L_{Atmos} - BW_{Cor} \quad (3)$$

where C_0 is the received power density at the receiver input in dBW/Hz, P_{TXo} is the transmitted power spectral density in dBW/Hz, G_{TX} is the transmit antenna gain at the high-power amplifier output in dB, G_{RX} is the antenna gain at the receiver input in dB, L_{FS} is the free space loss in dB, and L_{Atmos} is the atmospheric loss in dB. Interference power, I_0, is calculated similarly to (1) with an additional correction term, BW_{Cor}, included to correct for mismatched bandwidth overlap between the interfering transmitter and the victim receiver.

The simulation tool is designed to have configurable analysis parameters with support for ITU standard definitions such as [3] and [4] prioritized. Operational data is gathered over the course of a simulation and is consolidated into raw data and report files. The raw data file supports post-simulation validation efforts, while the report files summarize data outlining per receiver performance metrics, such as the percentage of contact time with interference.

2.2 Simulation Dimensions

Constellation parameter sensitivity sweeps offer the first dimension that can be broken out from the larger scenario for parallel computation. Due to the independent nature of these scenarios, computation can be completed independently for each scenario. These independent computations can then be split across multiple physical nodes, introducing horizontal scaling of the simulation and allowing the tool to complete large numbers of scenario permutations faster.

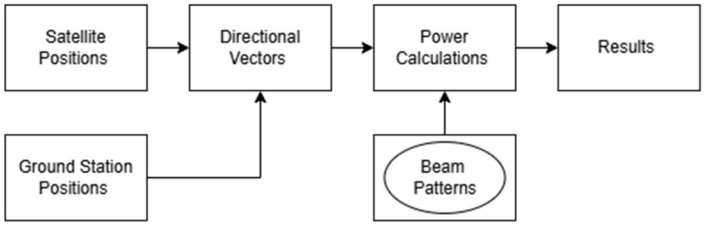

Figure 1. Simulation Parallelization Over Geographic Space Components

The second dimension to be broken out for parallel computations includes the geographic space components (see Figure 1). Projecting signal power from multiple variably located satellites down to ground stations shows strong potential for GPU acceleration due to the independent nature of these calculations. By building the time instances each as a large matrix representing a set of Earth-centered Earth-fixed (ECEF) coordinate satellites, the GPU can compute the projection of defined beam patterns to designated ground stations. The beam patterns serve as masks to the vectorized computations determining the signal power received from a satellite at a particular ground station.

The final dimension of parallelization is time (see Figure 2). Time is the most difficult factor to parallelize due to the mechanism utilized for selecting station contacts. A time-dependent contact mechanism is employed, in which the

selection mechanism chooses the longest contacts and iterates until no additional contacts are available to any satellites. This creates dependence on both past selections and a non-sequential search through time. The time dimension is handled once the schedule of contacts is determined. An additional dimension is added to the satellite positional data, allowing for larger matrix computations well suited to a GPU-based solution. This data must be masked by the contact schedule to ensure only transmitting satellites or stations are evaluated in the computations. Combining these parallelization methods is expected to significantly increase computation speed for large simulations.

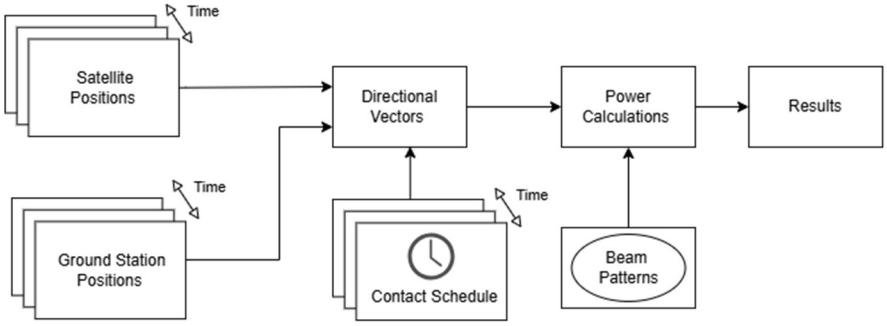

Figure 2. Simulation Parallelization Over Time

2.3 Libraries and Technologies

The simulation tool utilizes multiple Python libraries to enable easier access to scaling of resources. The key libraries utilized include PyTorch [10] and Ray [11]. PyTorch enables vertical scaling by computing large matrix calculations on GPUs. With native access to CUDA [12] and tensor cores on GPUs, PyTorch presents a strong prototyping platform for these simulations. To introduce horizontal scaling, Ray is utilized. Ray supports the execution of processes across multiple hosts with specific tasks queued for execution. The results are then pooled and analyzed or stored within a database. This analysis utilizes Pandas [13], a Python library with key capabilities in data analysis. Pandas provides the capability to interface with a PostgreSQL database for storage of results for later access.

Each stage of a simulation depends on data produced in the previous stage. Orbit propagation produces the satellite positions represented within a PyTorch tensor object. Then visibility masking is applied to identify what contacts can be made throughout the simulation timeline. From the contact availability, the selection algorithm chooses contacts and finalizes the schedule for the simulation. The finalized schedule acts as a mask to the later phase over the time dimension to select which assets are transmitting and where antennas are tracking. In the case of targeted satellite pointing, it also determines which station a satellite is directing the signal at. With computations for power completed across the constellation, the results are assembled and analyzed to produce summary figures on interference impacts.

The simulation tool is built based on EarthMesh™, an elastic geospatial and temporal simulation engine. This Calian-developed technology employs GPU acceleration for satellite communication computations. By utilizing an S2 geospatial indexing grid [14], EarthMesh™ can perform repeated calculations across the Earth's surface. Due to the nature of ground station-based computation where the S2 grid would not provide added benefit, the grid itself is not used within the simulations. The remaining portions of the EarthMesh™ product are utilized to improve the capabilities of the system. These capabilities include necessary link budget calculations such as beam projection, attenuation, beam EIRP, and carrier-to-interference ratios. Additionally, EarthMesh™ provides the GPU functionality required for high levels of parallelization.

2.4 Data Flow

The simulation tool utilizes matrix calculations for computations throughout the data flow from configuration until results are determined. Within the structure of PyTorch tensors, the simulator uses rows, columns, and additional dimensions to form the data sets. The first stages utilize Orekit for propagation and directly imports satellite ECEF position data into a tensor. This, combined with ground station locations converted into a tensor utilizing ECEF coordinates, forms the basis of the visibility calculations used to determine contact opportunities (see Figure 3). Both sets utilize a three-element coordinate system alongside an ID column and a time dimension.

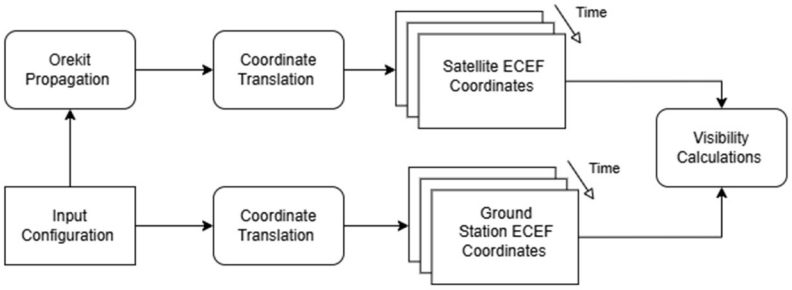

Figure 3. Visibility Assessment Flow Diagram

Visibility is calculated through CUDA computations within a GPU or on the central processing unit (CPU) as specified by the configuration of PyTorch to utilize GPU or CPU resources. These operations result in a contact opportunity schedule represented as a tensor with rows for each pass, columns for start time, end time, ground station, and satellite. The process of deconflicting filters this list of passes and creates a contact schedule (see Figure 4).

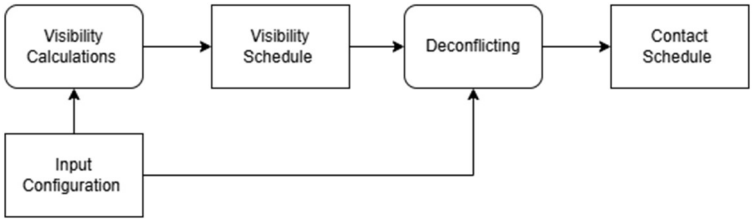

Figure 4. Contact Schedule Generation Flow Diagram

The contact schedule forms the basis of the power calculations alongside the tensors for satellites and ground stations. Based on antenna patterns and satellite and ground station configuration, the power received at terminals is computed between all visible ground stations and the satellites for the time instance. The contact schedule is then used to filter the tensor to only transmitters and receivers in operation at the time index. This is done to preserve the tensor structure through computation. Once the computations are completed, the outputs are sparse tensors for carrier power, interferer power, aggregate interference, carrier-to-interference, and aggregate carrier-to-interference. These tensors are used to generate a pass-based view of statistics. They result in a generated pass-indexed tensor that supports the statistics outputs for the reporting output. These results are then passed into a Pandas data frame utilized for final statistic generation. The tool then synthesizes the results into raw data and report files that outline the results of the simulation.

3. Results

The development of this tool resulted in two distinct investigation phases. The first investigation consisted of early inquiries into performance, while the second investigation considered improvements possible within the context of a complete simulation scenario. The results from the first investigation highlight improvements made through vectorization of individual computation steps, which showed between 10x-200x improvement in computation speed. Benchmarking for the initial investigation phase was completed with a constellation size of 1000 for a single time instance with 384 ground stations. The key results from this phase include:

- Vectorizing coordinate conversion reduced computation time of this stage from 164.7 ms to 0.4253 ms.
- Vectorizing the projection of satellites' positional vectors to every point in the simulation reduced computation time of this stage from 1901.9 ms to 11.92 ms.

The secondary phase focused on optimizations that impact a complete simulation scenario. The simulation scenarios evaluated for the benchmarking results utilize the parameters outlined in Table 1. The scenarios evaluated for collecting benchmarking results were performed in Calian's testbed environment utilizing one or two hosts matching the following specifications:

- **Processor**: AMD Ryzen 7700X, 8 physical cores, 16 threads, 4.5 GHz max
- **Memory**: Corsair Vengeance 64 GB (2x32) DDR5 5600
- **GPU**: ASUS TUF Gaming GeForce RTX 4090 OC Edition

Table 1. Benchmarking Simulation Parameters

	Quantity	Pointing Mode	Orbit Propagation Method	Operating Schedule	Minimum Elevation Angle
Victim Constellation	3	Tracking	Eckstein Hechler	-	-
Interfering Constellation	3 - 500	Tracking	SPG4	-	-
Ground Stations	6 - 73	Tracking	-	24/7	5˚ - 15˚

Improvements over multiple optimization phases were seen during the secondary investigation (see Table 2 and Table 3). The levels of optimizations are as follows:

- 0: No optimization, initial implementation.
- 1: Optimization of individual functions to improve parallel operations (i.e., inactive station filtering improvements).
- 2: Optimization of computation stages to support parallelized computation over time.

Table 2. Result Data - CPU

Simulation Run	Optimization Level	Satellites	Ground Stations	Time Frame	Granularity	Average Computation Time
1	0	6	6	1 day	30s	10s
2	1	6	6	1 day	15s	16s
3	0	6	6	7 days	30s	80s
4	1	603	66	1 day	30s	109s
5	2	203	73	1 day	30s	41.9s
6	2	303	73	1 day	30s	76.5s

Table 3. Result Data - GPU

Simulation Run	Optimization Level	Satellites	Ground Stations	Time Frame	Granularity	Average Computation Time
7	1	603	66	1 day	60s	44.2s
8	2	103	73	1 day	30s	12.3s
9	2	203	73	1 day	30s	18.5s
10	2	400	73	2 hours	5s	6.18s
11	2	500	73	2 hours	5s	7.8s

4. Discussion

The results show that increasing parallelization continues to significantly improve the speed at which simulation scenarios can be performed and evaluated. The computational speed improvements allow for simulations with an increased time granularity and longer duration to be evaluated in a shorter amount of time. Additional simulation speed improvements are anticipated with the implementation of a fully time-parallelized solution. Complete time-parallelization is blocked by the current contact mechanism that requires knowledge of past selections and performs non-sequential searches through time. Further investigation is required to achieve complete time-parallelization.

Limitations in video random access memory (VRAM) limit the size of simulations completed on GPUs, as shown in Table 3. The memory limitations encountered indicate that additional effort is required to reduce data requirements or to improve the ability to shift data quickly for computation between the host memory and VRAM of the GPU.

The results of this effort show a strong application in the wider satellite management domain. The flexibility of software-defined satellites can utilize the technology improvements here to promote an on-demand approach to resource management. Instead of a pre-planned model of resource usage, dynamic demand and calculation based on live environmental factors could be used to automatically update flexible satellite platforms to perform near-autonomous operations.

5. Conclusions

Given the resource constraints of spectrum availability, it is expected that interference mitigation will become an increasingly complex and dynamic activity. The need to share spectrum effectively will lead to the creation and adoption of spectrum leasing technology for flexible satellite platforms. The simulation technology presented here is capable of dynamically evaluating requests for interference determination considering lease time and orbital parameters to maximize satellite utilization while operating within spectrum constraints.

Beyond the gateway links simulated in this project, the application of this technology in user-based links for the satellite communications industry can provide valuable data in shortened time frames. The EarthMesh™ platform used in this project maintains the capability to utilize a global gridding system to provide regions of calculation that can be used for satellites with a large number of terminals that are generally geographically grouped. The interference calculations completed in this simulation are effective in planning spectrum distribution across a satellite network.

While ongoing research and refinement remain essential, initial results indicate that this highly parallel satellite network simulation solution will be a potent tool for a wide range of satellite communication applications. The results presented indicate significant improvement in computational speed in comparison to non-parallel simulation solutions. As an early technology, further development is required to fully realize the benefits of the simulation tool. Memory management and time series vectorization remain key future activities to actualize the benefits of the system. However, as constellations continue to expand in both flexibility and size, our approach promises to keep pace with the demands of our interconnected cosmos.

Acknowledgements

We would like to acknowledge Ron Osterried, Joshua DeJong and Cameron Oehler for their contributions to this project. The work described here was supported by the Canadian Space Agency Space Technology Development Program.

References

[1] N. U. L. Hassan, C. Huang, C. Yuen, A. Ahmad and Y. Zhang, Dense Small Satellite Networks for Modern Terrestrial Communication Systems: Benefits, Infrastructure, and Technologies, IEEE Wireless Communications, vol. 27, no. 5, pp. 96-103, October 2020.
[2] H. W. Jones, The Recent Large Reduction in Space Launch Cost, 48th International Conference on Environmental Systems, ICES-2018-81, Albuquerque, New Mexico, 2018, 8-12, July.
[3] Recommendation ITU-R SA.1027-6: Sharing criteria for space-to-Earth data transmission systems in the Earth exploration-satellite and meteorological-satellite services using satellites in low-Earth orbit, https://www.itu.int/rec/R-REC-SA.1027/en, August 2019, (accessed 01-03-25).
[4] Recommendation ITU-R SA.609-2: Protection criteria for radiocommunication links for manned and unmanned near-Earth research satellites, https://www.itu.int/rec/R-REC-SA.609/en, March 2006, (accessed 01-03-25).
[5] A. Yastrebova, A. Anttonen, M. Lasanen, M. Vehkaperä and M. Höyhtyä, Interoperable Simulation Tools for Satellite Networks, 2021 IEEE 22nd International Symposium on a World of Wireless, Mobile and Multimedia Networks, Pisa, Italy, 2021, pp. 304-309.
[6] F. Daneshvar, Comparison of satellite networks simulators for mega-constellations, M.S. thesis, Politecnico di Torino, Turin, Italy, 2024.
[7] Orekit, 15 June 2024, https://www.orekit.org/, (accessed 01-03-25).
[8] Recommendation ITU-R P.525-4: Calculation of free-space attenuation, https://www.itu.int/rec/R-REC-P.525/en, August 2019, (accessed 01-03-25).
[9] Recommendation ITU-R P.676-10: Attenuation by atmospheric gases, https://www.itu.int/rec/R-REC-P.676-10-201309-S/en, September 2013, (accessed 01-03-25).
[10] PyTorch, https://pytorch.org/, (accessed 01-03-25).
[11] Ray, https://www.ray.io/, (accessed 01-03-25).
[12] CUDA Toolkit, https://developer.nvidia.com/cuda-toolkit, (accessed 01-03-25).
[13] Pandas, https://pandas.pydata.org/, (accessed 01-03-25).
[14] S2Geometry, http://s2geometry.io/devguide/s2cell_hierarchy.html, (accessed 21-02-25).