[CP-02-002] High Throughput Computing and GIS

Abstract

High Throughput Computing (HTC) is a computing paradigm that focuses on maximizing the number of tasks completed over time by distributing independent computational jobs across multiple systems. Unlike High Performance Computing (HPC), which prioritizes speed and raw processing power, HTC emphasizes throughput, making it ideal for workloads involving numerous independent tasks. This approach is widely used in GIScience, where researchers handle diverse datasets and computationally intensive simulations. HTC operates by executing multiple tasks in parallel without requiring inter-process communication. It leverages distributed computing resources, such as computer clusters, grids, and cloud-based platforms, efficiently utilizing idle processing power. This allows for the simultaneous processing of vast datasets, significantly reducing computation time. In GIScience, HTC proves valuable for environmental data analysis, climate modeling, satellite image processing, and geological simulations. Researchers benefit from its scalability, fault tolerance, and flexibility, enabling them to accommodate a wide range of computational tasks. The ability to scale resources as needed and maintain computation despite individual task failures makes HTC a robust tool for scientific advancements.

Author & citation

Carbajales-Dale, P., Cleveland, S., and Post, G. (2025). High Throughput Computing and GIS. The Geographic Information Science & Technology Body of Knowledge (2025 Edition), John P. Wilson (ed.). DOI: 10.22224/gistbok/2025.1.1.

Explanation

1. Definitions
2. Introduction
3. Core Technologies and Architectures
4. Relevance of HTC in GIS
5. Challenges and Solutions
6. Future Trends in HTC for GIS

1. Definitions

High Throughput Computing (HTC): A computing paradigm that utilizes distributed computing resources to execute a large number of independent jobs concurrently, optimizing the volume of tasks processed over time.

High Performance Computing (HPC): A computing paradigm that uses parallel processing techniques to solve complex problems requiring significant computational power and the simultaneous use of many processors.

CPU: A Central Processing Unit (CPU), is the primary component of a computer responsible for executing instructions and performing calculations. It is often referred to as the “brain” of a computer, as it coordinates the activities of other hardware components. Modern CPUs

contain multiple cores, allowing them to perform multiple tasks simultaneously, thereby enhancing overall performance.

Serial Processing: A computing approach of handling tasks in a sequential order, one after the other, on a single processor or core. Each task must be completed before the next one begins, without overlapping, which can be slow for large or complex computations.

Parallel Processing: A computing approach where multiple tasks or parts of a task are executed simultaneously across multiple processors or cores within a single computer or across multiple computers.

Distributed Processing: A computing approach where computational tasks are divided and distributed across multiple computers or nodes connected via a network. Each node processes a portion of the task independently, and results are combined to produce the final output.

Cluster: A set of interconnected computers that work together as a single system to perform tasks (or jobs). These computers, often referred to as nodes, can be physical or virtual machines, and they are networked together to share resources such as processing power, memory, and storage.

Processor: A processor, also known as Central Processing Unit (CPU), is the primary component of a computer responsible for executing instructions and performing calculations. The processor's primary functions include interpreting and processing data, as well as managing data flow between the computer's memory, input/output devices, and other components.

Scheduler: A system software component that manages the allocation and execution of tasks (or jobs) on computing resources.

Middleware: Software that acts as an intermediary between different systems, applications, or components, enabling them to communicate and interact with each other.

HTCondor: An open-source HTC software system designed to manage and distribute large computational workloads across a distributed network of computing resources (Livny et al., 1997). Originally developed by the Condor Project, a research program led by Miron Livny at the University of Wisconsin, HTCondor has been supported and maintained by the HTCondor team for over two decades (Bricker et al., 1992). It supports multiple operating systems, including Windows, Linux, and macOS, making it a popular choice amongst GIS researchers.

2. Introduction

High Throughput Computing (HTC) is a computing approach designed to partition large workflows into a series of distributed computing tasks running simultaneously. This paradigm aims to maximize the number of tasks completed over a given period, making it particularly suited for applications requiring numerous independent tasks. Unlike High Performance Computing (HPC), which is optimized for raw processing power and speed, HTC is characterized by its ability to process a high volume of computational jobs concurrently, maximizing throughput over a period of time. This approach is particularly beneficial in fields such as GIScience, where researchers often encounter diverse datasets

and complex simulations that require extensive computational resources.

The essence of HTC lies in its ability to manage and execute multiple computational tasks that do not need to communicate with each other during their execution. This independence is key to HTC's efficiency, as it allows for the distribution of tasks across a grid or cluster of computers, often leveraging idle processing power in a network of workstations or across cloud-based resources. By doing so, HTC can handle a vast array of jobs simultaneously, significantly reducing the time required to obtain results.

In the context of GIScience, HTC can be instrumental in processing environmental data, such as determining invasive species threats (Watkins, 2021), climate modeling (Clare, 2019), analyzing satellite imagery (Zhang et al., 2022), and simulating geological processes (Fienen & Hunt, 2015). These applications often involve the manipulation of large-scale datasets and the execution of numerous independent tasks, making HTC an ideal fit. The ability to run these tasks in parallel, without the need for inter-process communication, enables GIS scientists to achieve higher productivity and make more timely discoveries.

Key characteristics of HTC include its scalability, fault tolerance, and flexibility. Scalability ensures that as the demand for computational power increases, the HTC system can expand by adding more resources without significant changes to its infrastructure. Fault tolerance is critical in HTC environments, as the failure of individual tasks does not impede the progress of others, ensuring that the overall computation can continue uninterrupted. Flexibility allows for a wide range of tasks to be accommodated, regardless of their computational requirements, making HTC a versatile tool for researchers.

Understanding HTC is crucial for GIS scientists as it opens new possibilities for research and exploration. By harnessing the power of HTC, researchers can process and analyze data at scales previously unattainable, leading to deeper insights and more accurate models of the Earth's processes. As such, HTC stands as a cornerstone of modern cyberinfrastructure, enabling scientific advancements through its unique approach to distributed computing.

1.1 Comparing HTC and HPC: Key Differences and Distinctions

High throughput computing (HTC) and high performance computing (HPC) are two paradigms that, while often conflated, serve distinct purposes in the realm of computational tasks. HTC is designed to process a large number of relatively small or independent computational jobs over a long period, optimizing for overall throughput rather than the speed of any single task. It is particularly well-suited for problems that can be parallelized without the need for inter-process communication. In contrast, HPC is engineered to tackle highly complex computational problems that require significant processing power and often rely on tightly coupled, communicative processes. HPC systems are characterized by their ability to perform a high number of floating-point operations per second (FLOPS), making them ideal for simulations, modeling, and other tasks that demand real-time or near-real-time processing speeds.

The architecture of HTC systems is typically more distributed and flexible, allowing for a variety of computational resources, such as idle desktops or cloud-based services, to contribute to the processing of jobs. This distributed nature also means that HTC environments can be more fault-tolerant, as the failure of one node does not necessarily impede the progress of other tasks in the queue. On the other hand, HPC environments are often composed of supercomputers or clusters with high-bandwidth interconnects and

advanced processing capabilities, necessitating a more centralized and controlled infrastructure.

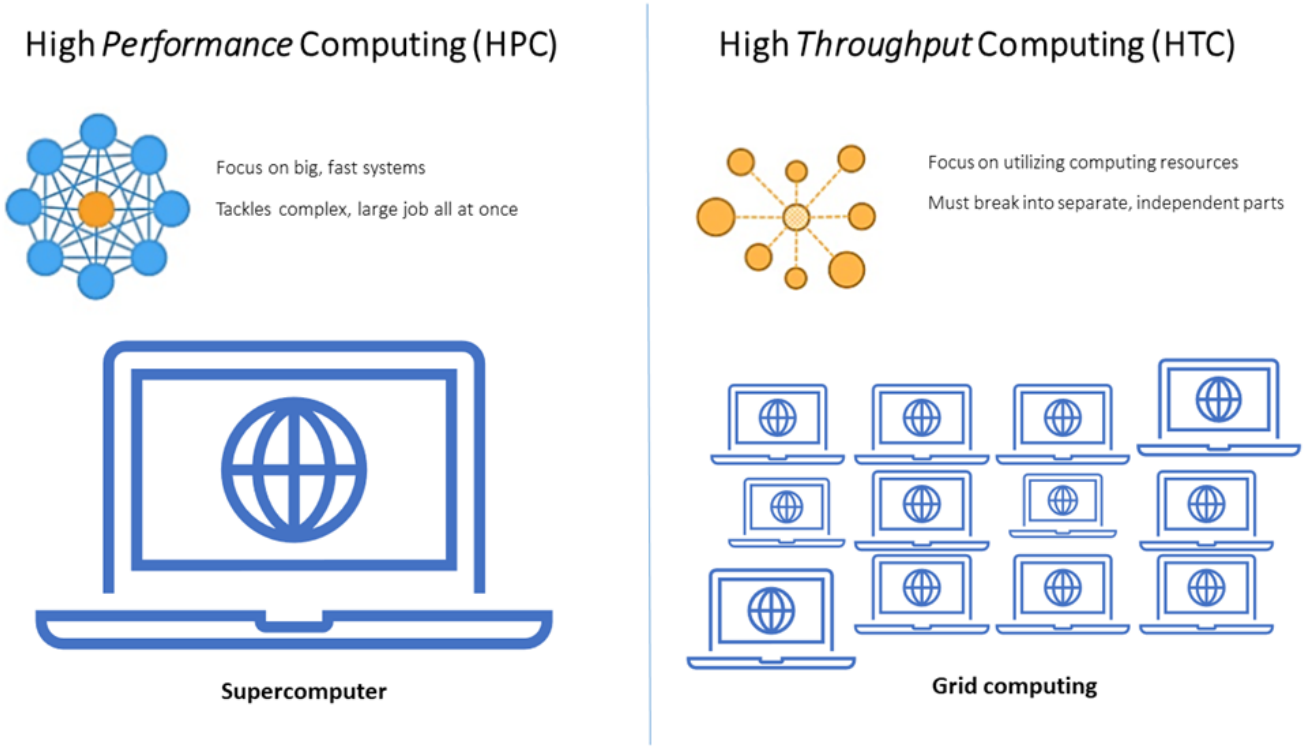


Figure 1. Comparing High Throughput Computing (HTC) versus High Performance Computing (HPC). Source: author.

In terms of application, HTC is frequently employed in fields such as bioinformatics, where researchers may need to run a multitude of independent simulations or data analyses. HPC, however, is commonly utilized in disciplines like atmospheric science or physics, where complex, interdependent calculations are required to accurately model systems or phenomena. The choice between HTC and HPC ultimately hinges on the specific requirements of the computational task at hand, with considerations for the scale, complexity, and urgency of the problem guiding the decision.

Moreover, the scheduling and management of tasks differ significantly between the two. HTC systems often use job schedulers that can handle a large number of queued jobs, managing them so as to maximize the utilization of available resources over time. HPC systems, conversely, prioritize the efficient execution of a smaller number of highly resource-intensive tasks, with schedulers that allocate resources to minimize the time to completion for each job.

While both HTC and HPC are critical components of modern cyberinfrastructure, they cater to different computational needs. HTC excels in maximizing the number of completed jobs over time for a wide array of independent tasks, making it the go-to choice for many research and industry applications that require such capabilities. This makes HTC particularly suitable for applications such as traffic analysis, where processes like calculating daily traffic counts along a road network can be divided into jobs that perform spatial intersections and aggregations independent from each other (Claflin, 2016).

In contrast, HPC excels in tasks requiring significant computational power and tightly coupled processing, such as urban growth simulations. These simulations often integrate diverse and extensive historical datasets for long-term forecasting. HPC's parallel processing architecture enables the distribution of complex calculations across multiple processors, making it indispensable for handling the computational intensity and interdependencies inherent in such models (Pijanowski et al., 2014). HPC, with its focus on speed and power, remains indispensable for solving some of the most challenging and complex problems in science and engineering today. Understanding the distinctions between these two computing paradigms is essential for researchers and practitioners in making informed decisions about the appropriate computational strategies for their specific projects.

2. Core Technologies and Architectures

2.1 Hardware and software architectures commonly used in HTC

The core technologies and architectures in HTC are pivotal for its effective operation. At the heart of HTC are distributed systems that leverage a variety of computational resources, often in a non-dedicated fashion. This means that HTC systems can utilize idle computing power across a network, which may include personal computers, servers, or even cloud-based resources.

The hardware architecture of HTC systems is typically characterized by a large number of compute nodes, which can be either homogeneous or heterogeneous in nature. These nodes are connected through a high-bandwidth network to facilitate communication and data transfer. In many cases, the nodes are standard off-the-shelf servers that are optimized for cost-efficiency and scalability rather than peak performance. The software architecture, on the other hand, is designed to manage and schedule tasks across these distributed resources efficiently. It includes middleware that can handle job queuing, distribution, monitoring, and error recovery. One of the most widely used software systems for HTC is HTCondor [Liztkow et al, Basney et al] which excels in managing job queues and efficiently distributing tasks across available resources.

HTC setups also employ various grid computing technologies, allowing them to span across multiple administrative domains and leverage resources from different organizations. This is facilitated by software that can negotiate resource sharing agreements and handle security and accounting, ensuring that each participant's policies are respected. The Open Science Grid (OSG) [Pordes et al, Sfiligoi et al, OSG 2006, OSG 2015] is a prominent example of such a collaborative effort, providing shared computing and data resources for scientific research. In the context of GIS, researchers have utilized OSG resources to generate habitat suitability maps for over 600 species in California and overlay these maps with wildfire extent data at a 30-meter resolution. This research, which would have been prohibitively time-consuming and computationally intensive on a desktop computer, was successfully completed by leveraging the computing capacity of more than fifty partners in the Open Science Grid pool (Ayars et al., 2023; Goeking, 2024)

In terms of data management, HTC environments often integrate with big data technologies to handle the vast amounts of data generated and processed by HTC tasks. This includes data virtualization and storage systems that can scale to accommodate the data-intensive nature of HTC workloads. Furthermore, HTC systems are increasingly incorporating cloud

technologies, offering flexible and dynamic resource provisioning. This fusion of HTC with cloud environments allows for a more adaptable and resilient infrastructure, capable of handling varying workloads with ease.

The user interfaces in HTC systems are typically designed to be user-friendly, providing researchers with tools to submit and manage their jobs without needing to understand the underlying complexities of the distributed system. This is complemented by automatic workflow management systems that can orchestrate the execution of tasks based on predefined criteria, further simplifying the user experience.

Overall, the hardware and software architectures commonly used in HTC setups are designed to provide a robust, scalable, and cost-effective solution for processing a high volume of computational tasks. By leveraging distributed resources and advanced scheduling algorithms, HTC systems can deliver significant computational throughput, facilitating research and discovery across various scientific disciplines.

2.2 Understanding distributed computing, grid computing, and cloud computing in HTC

Distributed computing is one of the foundational technologies for HTC. It involves a network of interconnected computers that work together to perform a task. This architecture allows for tasks to be executed concurrently across different machines, optimizing resource utilization and reducing completion time. Distributed systems can be scaled horizontally, adding more machines as needed, which is ideal for HTC applications that require significant computational power.

Grid computing is a specialized form of distributed computing that involves the integration of heterogeneous resources, often spanning multiple administrative domains. It is designed to support complex, resource-intensive tasks that cannot be handled by a single computer or local network. Grid computing systems are characterized by their ability to provide a high level of control over resources, ensuring that tasks are executed in an environment that meets specific performance criteria.

Cloud computing has emerged as being transformative in HTC, offering on-demand access to a scalable pool of shared computing resources. Cloud services can be rapidly provisioned and released with minimal management effort, providing a flexible and cost-effective solution for HTC tasks. The pay-as-you-go model of cloud computing aligns well with the sporadic or variable demand seen in some HTC workloads.

The interplay between these core technologies enables HTC to thrive. Distributed computing lays the groundwork for task execution across multiple nodes, while grid computing extends this capability to more diverse and powerful resources. Cloud computing adds a layer of flexibility and scalability, allowing HTC practitioners to dynamically adjust to the computational demands of their projects. Together, these technologies form a robust infrastructure that supports the diverse and evolving needs of high-throughput computing.

3. Relevance of HTC in GIS

In recent decades, the world has faced significant societal challenges, including disease outbreaks, natural disasters, growing social inequality, and climate change. These global

problems are inherently geospatial, making the role of GIS scientists crucial in addressing them (Li, 2020). Simultaneously, advancements in technology and data generation processes have led to an exponential increase in the volume of geospatial data. The rapid commoditization of earth observation systems, the expansion of wireless sensor networks, and the surge in geospatial big data, such as Point of Interest (POI) data and social media feeds, have contributed to an extraordinary growth in the availability and detail of geospatial data (Li et al., 2020).

While the emergence of geospatial data at unprecedented spatial, temporal, and spectral resolutions has enabled GIS researchers to better understand and address these societal issues, it also presents a significant challenge: effectively analyzing this vast amount of data. For instance, in hydrological modeling, advances in digital photogrammetry and remote sensing technologies have produced high-resolution terrain datasets that allow for the extraction of drainage networks with remarkable detail. However, traditional optimization methods are unable to process such large terrain datasets in a serial computing mode.

To address this limitation, researchers at Wuhan University leveraged HTC to improve the efficiency of processing large Digital Elevation Models (DEMs) (Gong & Xie, 2009). Their HTC setup consisted of a Condor pool utilizing six available workstations connected through a local area network (LAN). Employing several spatial techniques to split the large DEM into smaller units, the HTC pool was able to extract drainage networks from high-resolution DEMs four times faster than serial computing methods. This research demonstrated that HTC workflows can process high-resolution data more efficiently, reducing computation time and enhancing the extraction of drainage networks.

In the fields of ecology and conservation, researchers have often had to compromise between the extent of their analysis and the granularity of their data due to limitations in computing capacity, which has subsequently limited the interpretation and applicability of their studies (Arponen et al., 2012). Moreover, disciplines such as land use planning increasingly demand greater computational capacity to enhance multi-objective planning. This enhancement is achieved by enabling systematic iterative modeling and sensitivity analyses based on complex spatial allocation problems (Janssen et al., 2008).

To demonstrate the feasibility of HTC in addressing these issues, Leonard et al. (2014) conducted several experiments in landscape and conservation planning. HTC was able to reduce processing time by nearly 100-fold. This reduction was crucial not only for efficiency but also for enabling iterative modeling at different scales and testing various modeling assumptions. The high-resolution of the data allowed the researchers to identify 4,600 potential new wetlands and significantly reduce human error, which is often encountered when planners need to manually examine thousands of aerial photographs (Leonard et al., 2014).

In summary, HTC offers significant advantages in GIS research and applications. It enables the use of higher resolution data and greater volumes of data over larger spatial extents, while also reducing human error and increasing computational accessibility. HTC can leverage existing computer infrastructure, making it a viable option for individuals and organizations with limited resources to access high-capacity systems. Its scalability and efficiency allow researchers to conduct more comprehensive and accurate analyses. These capabilities are essential for addressing complex geospatial problems and, consequently,

for tackling critical societal challenges.

4. Challenges and Solutions

The integration of HTC in GIScience presents a multifaceted landscape of challenges and potential solutions. As GIS continues to evolve with the increasing availability and complexity of geospatial data, HTC emerges as an essential tool for processing large geospatial datasets. However, several significant challenges must be addressed to fully leverage HTC in GIS research applications.

One of the primary challenges in utilizing HTC for GIS is the division and subsequent merging of large spatial datasets. While HTC excels in processing multiple independent jobs concurrently, simply dividing large datasets into smaller spatial units is not always an optimal solution. This approach can lead to issues with data consistency, redundancy, and most importantly, the integrity of spatial relationships. For example, in the case of extracting drainage networks from large terrain datasets, Gong and Xie decomposed the large DEM using natural watershed boundaries derived from multi-scale DEMs (Gong & Xie, 2009). Furthermore, merging these smaller units back into a coherent dataset poses additional challenges, such as ensuring data accuracy, integrity, and handling discrepancies that arise during the division process. To address these issues, advanced algorithms and techniques are required for efficient data partitioning and merging, ensuring that the integrity and quality of the geospatial data are maintained.

HTC is typically implemented using existing grids, such as GIS-enabled university computer labs, making it a more cost-effective and accessible option compared to HPC, which is typically carried out on supercomputers. However, setting up an HTC environment still requires a baseline level of IT support for installation, configuration, and ongoing management. Organizations might encounter challenges in allocating the necessary IT resources and expertise to support these activities. To mitigate these challenges, it is important IT personnel are adequately trained in managing HTC environments. This can be achieved through review of documentation, active participation in user groups, and staying informed about bug fixes and upcoming software releases.

Another challenge is the reticence of the GIS community towards non-GUI software packages necessary for utilizing HTC environments. Many GIS researchers are accustomed to proprietary software solutions, whether desktop or cloud-based, and may be hesitant to transition to a programming environment. To overcome this barrier, it is essential to provide comprehensive training and demonstrate the robustness and advantages of using HTC for their analyses. Additionally, fostering a collaborative community around programming languages such as Python for GIS and HTC tools can help build confidence and encourage wider adoption.

Despite these challenges, the benefits of using HTC in GIScience are substantial. HTC enables the cost-effective use of higher resolution data, greater volumes of data, and broader spatial extents, while also reducing human error and increasing computational accessibility. It is both scalable and efficient, allowing researchers to conduct more complex and accurate analyses. By addressing the challenges related to data division and merging, IT support, and community reticence, GIScience can fully harness the power of HTC to tackle complex geospatial problems and contribute to solving pressing societal challenges.

5. Future Trends in HTC for GIS

Despite the exponential growth in data processing capabilities, the analysis of increasingly fine-grained datasets across broader spatial extents remains a significant computational challenge, often exceeding the capacity of many researchers, particularly in the face of rising cloud-computing costs. HTC offers a viable alternative by leveraging existing computational infrastructure, making it an accessible option for individuals and organizations with limited financial resources to invest in high-capacity CPUs.

The flexibility and scalability of HTC allows for researchers and institutions to combine existing resources with minimal investment in hardware infrastructure and offer dynamic solution to the computational needs of the GIS research community. This flexibility in capacity is particularly advantageous as GIS workflows increasingly shift toward real-time data analysis, incorporating machine learning and artificial intelligence for applications such as disaster response, environmental monitoring, predictive analytics, automated feature extraction, and pattern recognition.

As GIS continues to evolve, the role of HTC in supporting these advanced workflows will become increasingly critical. By providing a cost-effective, scalable solution that can dynamically meet the computational demands of modern GIS research, HTC is poised to drive significant advancements in the field, enabling more comprehensive and timely geospatial analyses.

References

Arponen, A., Lehtomäki, J., Leppänen, J., Tomppo, E., & Moilanen, A. (2012). Effects of Connectivity and Spatial Resolution of Analyses on Conservation Prioritization across Large Extents. Conservation Biology, 26(2), 294–304.

Ayars, J., Kramer, H. A., & Jones, G. M. (2023). The 2020 to 2021 California megafires and their impacts on wildlife habitat. Proceedings of the National Academy of Sciences, 120(48), e2312909120.

Bricker, A., Litzkow, M., & Livny, M. (1992). Condor Technical Summary, Version 4-1b. Computer Sciences Department, University of Wisconsin - Madison.

Claflin, P. (2016, August 24). When a lot of little things add up to a lot of time; HTCondor to the rescue. ACI-REF Blog post.

Clare, R. (2019, January 8). Running Numerous WRF Simulations Simultaneously with High Throughput Computing. 99th American Meteorological Society Annual Meeting. Phoenix, Arizona.

Fienen, M. N., & Hunt, R. J. (2015). High-Throughput Computing Versus High-Performance Computing for Groundwater Applications. Groundwater, 53(2), 180–184.

Goeking, B. (2024, February 26). Ecologists utilizing HTC to examine the effects of megafires on wildlife. Blog post. The OSG Consortium.

Gong, J., & Xie, J. (2009). Extraction of drainage networks from large terrain datasets using

high throughput computing. Computers & Geosciences, 35(2), 337–346.

Janssen, R., van Herwijnen, M., Stewart, T. J., & Aerts, J. C. J. H. (2008). Multiobjective Decision Support for Land-Use Planning. Environment and Planning B: Planning and Design, 35(4), 740–756.

Leonard, P. B., Baldwin, R. F., Duffy, E. B., Lipscomb, D. J., & Rose, A. M. (2014). High-throughput computing provides substantial time savings for landscape and conservation planning. Landscape and Urban Planning, 125, 156–165.

Li, W. (2020). GeoAI: Where machine learning and big data converge in GIScience. Journal of Spatial Information Science, 2020(20), 71-77.

Li, W., Batty, M., & Goodchild, M. F. (2020). Real-time GIS for Smart Cities. International Journal of Geographical Information Science, 34 (2), 311-324.

Livny, M., Basney, J., Raman, R., & Tannenbaum, T. (1997). Mechanisms for high throughput computing. SPEEDUP Journal, 11(1), 36-40.

OSG Consortium. (2006). OSPool. The OSG Consortium.

OSG Consortium. (2015). Open Science Data Federation (OSDF). The OSG Consortium.

Pijanowski, B. C., Tayyebi, A., Doucette, J., Pekin, B. K., Braun, D., & 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, 250–268.

Pordes, R., Petravick, D., Kramer, B., Olson, D., Livny, M., Roy, A., Avery, P., Blackburn, K., Wenaus, T., Würthwein, F., Foster, I., Gardner, R., Wilde, M., Blatecky, A., McGee, J., & Quick, R. (2007). The open science grid. Journal of Physics: Conference Series, 78, 012057.

Sfiligoi, I., Bradley, D. C., Holzman, B., Mhashilkar, P., Padhi, S., & Wurthwein, F. (2009). The pilot way to grid resources using glideinWMS. 2009 WRI World Congress on Computer Science and Information Engineering, 2, 428–432.

Watkins, J. (2021, November 9). Protecting ecosystems with HTC. Blog post. The OSG Consortium.

Zhang, S., Xue, Y., Zhou, X., Zhang, X., Liu, W., Li, K., & Liu, R. (2022). State of the Art: High-Performance and High-Throughput Computing for Remote Sensing Big Data. IEEE Geoscience and Remote Sensing Magazine, 10(4), 125–149. IEEE Geoscience and Remote Sensing Magazine.