SECOND GENERATION PROTOTYPE OF A DESIGN PERFORMANCE OPTIMIZATION FRAMEWORK

VOLKER MUELLER
Bentley Systems, Incorporated, Chapel Hill, NC, USA
Email address:volker.mueller@bentley.com

Abstract. The integration of performance evaluation into the building design process becomes increasingly important in order to respond to demands on contemporary design with respect to the future of our built and natural environments. This paper presents work on the second iteration of an implementation of a design performance optimization framework that attempts to respond to the challenges of integrating analysis and optimization into the design process. Main challenges addressed are speed of feedback through implementation on the cloud, utilizing parallelization of computations and availability of results in the computational context of the model through leveraging the parametric nature of the application; The goal is to enable designers in their decision-making throughout the design process with focus on earlier phases of the design process during which changes can be implemented faster and at much lower costs than in later phases of design or even during construction and occupation.

1. Introduction

This paper presents work on the second implementation iteration (S2I) of a design performance optimization framework that attempts to respond to as many of the following challenges as possible, which are described in more detail in a separate paper by Mueller et al. (2013). The differentiation in the challenges is necessary to guide the research in the right direction for finding solutions and reducing the risk of addressing problems where they cannot be resolved.

(1) Interoperability: the building design software industry is similarly fragmented as the building industry at large. There are many incompatible software programs and data formats. How is the interdisciplinary collaboration supported that is necessary for successful high-performance buildings?
(2) Data discrepancy: conceptual design is less concerned with the detailed information required by analysis software (Bleil de Souza 2012). How can the designer escape the resulting limitations in analysis opportunities during early design?
(3) Data equivalency: design authoring tools may not allow creation of all data required by analysis tools, leading to gaps in data that prevent successful analyses. How can software deal with these gaps?
(4) Speed of feedback: Design is an iterative process. Iterations in design need to be frequent and fast (Hetherington et al. 2011). How can design feedback be kept fast enough to remain relevant for the current iteration?

(5) Performance proxies: which simplified analysis methods are available or are there simple analyses or indicators of a different type that may support prediction of future performance?
(6) Results display: visualization of analysis results is not related to the geometry model (Dondeti and Reinhart 2011). How can a designer be effectively enabled to understand the performance of the design in relation to the design components responsible for the respective performance?
(7) In-context results: analysis results are not available in the digital model in a way that permits computations to be performed on them. How can automation of refinement iterations or multi-objective optimization routines be enabled in an effective manner?
(8) Human-machine balance: not all design goals are measurable. How are "hard" computed performance metrics balanced with "soft" qualitative aspects?

In the implementations we focused on addressing two of these challenges. The first is the speed of feedback which is being addressed by implementing the system on the cloud providing access to virtually unlimited computing resources, thus allowing accelerating optimization processes through parallelization. The second is in-context results which are required for setting up an optimization loop. While in-context results benefit the user in other use scenarios, as well, they do need to be available associated with the respective elements of the design for optimization processes so that subsequent fitness computations can be executed with sufficient specificity in order for stochastic exploitation of parameter to fitness dependencies by the optimization algorithm. On the way we had to address some of the other challenges, especially interoperability and data discrepancy. Additional challenges like results display and human-machine balance are inherently addressed in the systems architecture and workflow. Data equivalency and performance proxies have not been resolved in these implementations.

A first prototype implementation of a software system responding to these challenges was introduced and tested at the Smartgeometry 2012 event in Troy, NY. In the following sections we will briefly describe the prototype and what we learned from it and how we proceeded with the second iteration of the implementation of this system. We describe benefits that have been already observed, and work that remains, as well as, conclusions that we have drawn from this process.

2. Proposed responses to these challenges

The general approaches to addressing the challenges remained the same for the first prototype and the S2I:
The currently dominant proposals for overcoming the interoperability challenge to achieve virtually seamless collaboration between design partners are (a) confining teams to a tight, closed proprietary system of software applications, unified through a proprietary data format; (b) a loose system of many software tools with individual translation modules on an as-needed basis resulting in many one-to-one mappings (Janssen et al. 2012); and (c) a loose system of tools using an open or otherwise published, standardized data format like the Industry Foundation Classes (IFCs). Latter two approaches frequently employ workflow control systems to connect all the modules required for a specific workflow (Flager et al. 2008; Toth et al. 2012). This research project uses a mix of the first and third approaches by using a shared data representation while initially focusing on the implementation of

frameworks that will allow loose association of authoring and analysis tools through an application programming interface (API).

We implement an analytic framework as the intermediary between authoring tools and analysis engines to overcome the challenge of data discrepancy. As broker between consumers of analysis services and those services, the analytic framework has the potential to point out any data discrepancies and leave it to the analysis services consumer either to amend the data or to select an analysis service for which sufficient data are available. A complementary approach is the search for analysis types that require less information in order to reduce occurrence of data discrepancy. Although this is a separate research project, since Smartgeometry 2012 an insolation analysis module has been added to the system to test this premise.

Neither the prototype nor S2I have focused on results display, yet. However, due to the innate capabilities of the parametric design authoring platform, various experiments were conducted to explore future display methods and styles. In support of these explorations capabilities have been added to S2I described later in this report. There is additional research and development work under way to respond to specific challenges of results display posed by the proposed system, about which will be reported separately. Similarly did the implementations not specifically address the balance between human control and delegation of decision-making to the machine. We leverage the inherent capabilities in the design authoring tool that permit the design team to design and develop models, designs, parameter choices, analysis choices, any post-processing and subsequent behaviours, and are accessible and changeable at every step of the process (Mueller et al. 2013)

The two challenges of design analysis integration and optimization that are most relevant to this phase of the research and for the implementations are the speed of feedback and availability of in-context results. Analysis results are returned through the analytical framework to the design authoring tool, where they are processed and presented so that they can be easily matched with the nodes to which they apply, either by the way they are structured, or through clear reference by node name. This does not only support results display on the analyzed geometry, but also additional computations, for example model changes that are geared towards improving the design's performance. However, most relevant in the context of the implementation is that in-context results allow computing fitness values for the optimization algorithm, while the parametric design system also provides for the capability for the optimization framework to manipulate design parameters to instantiate variations of the design in order to find better performing solutions.

Speed of feedback is at the core of the motivation for this research and development project. Analysis processes are time consuming. Even if they execute within a few minutes or just dozens of seconds, in the perception of designers immersed in a design process any such time is too long and disruptive. In optimization processes, these times are multiplied depending on the optimization algorithm in use, making optimization even less feasible. Therefore, a cloud implementation promises to permit parallelization of processes to reduce feedback time into a more feasible time range.

3. First prototype implementation for Smartgeometry 2012

The software architecture of the first prototype implementation for Smartgeometry 2012 is shown in Figure 1.

The prototype uses as design authoring tool GenerativeComponents (GC) from Bentley Systems [1]. GC is a parametric dependency and relationship modelling system. Its object classes are called *node types*, and the class instances are *nodes* in a diagrammatic graph with some nodes also generating a geometric representation in a geometric model view. Dependencies and relationships are represented in the graph by connecting curves, in the following called *wires*. The relationship graph in GC is a directed, acyclic graph (DAG). The user interacts with GC through the geometric model, the graph representation, or through scripting. All three modes of interaction are supported as representations of the underlying parametric model and can be used interchangeably and concurrently.

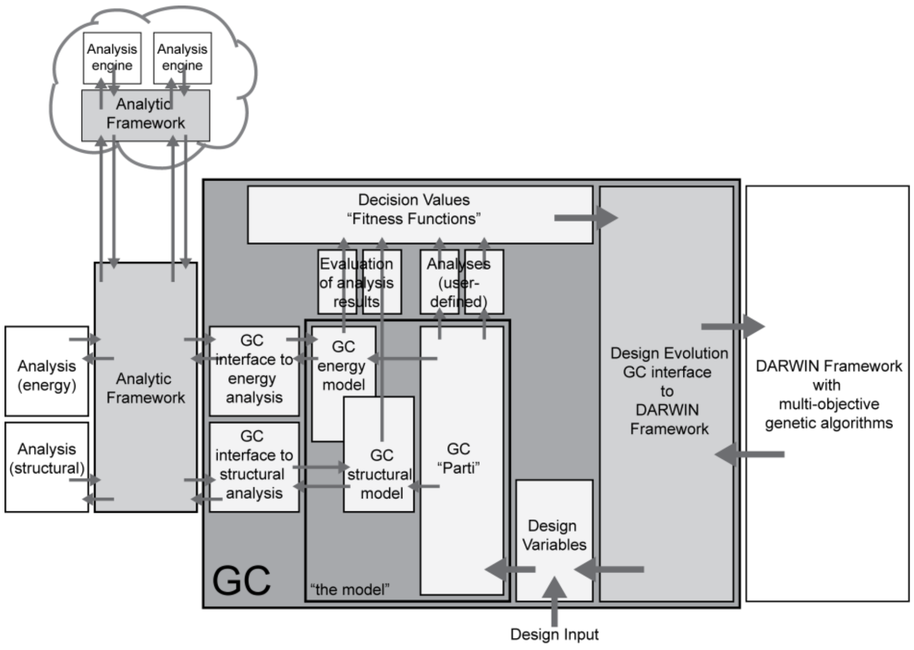
Analysis engine
Analysis engine
Analytic Framework
Decision Values "Fitness Functions"
Evaluation of analysis results
Analyses (user-defined)
Analysis (energy)
Analysis (structural)
Analytic Framework
GC interface to energy analysis
GC interface to structural analysis
GC energy model
GC structural model
GC "Parti"
Design Evolution GC interface to DARWIN Framework
DARWIN Framework with multi-objective genetic algorithms
Design Variables
GC
"the model"
Design Input

Figure 1. Software architecture and process flow of the prototype implementation.

As structural analysis service Bentley Systems' STAAD engine is used [2]. For energy analysis the system uses the Department of Energy's EnergyPlus engine [3]. In order to generate a coherent design and analysis model, special GC node types for structural and energy entities were implemented, which can be modelled with dependencies and relationships to each other, so that construction of a consistent model is straightforward. We chose these node types to provide data for each of the analysis engines as closely to the required minimum as possible. There are a few shared node types, like the project node type providing project-level information like location, building type, and climate data; or the material node type, allowing either selection of a few predefined materials or user defined materials with properties as required. If users utilize only one of the analysis types they do not need to enter the properties required by other analysis types.

The analysis engines have their footprint in GC as analysis node types. Users wire only those nodes that they want analyzed into the analysis nodes. On the analysis nodes users may switch between local or cloud-based execution of the analysis. Once the analysis is triggered from the node, analytical data are being generated, STD files for STAAD, IDF files for EnergyPlus, and a few supplemental files as needed. Depending on the execution mode, the client-side analytical framework passes the job to the analysis engine services installed and running on the client machine, or it passes the job to the cloud. We are using Microsoft's Azure as cloud platform. In order to access those resources, users need to authenticate through an authentication service. Then the analytic framework uses file upload services to transmit the job data to the cloud where the cloud-side counterpart of the analytic framework receives the job request and passes it to the processing queue which dispatches the jobs to the available nodes.

The analysis engines compute the various simulations and generate results files that may be post-processed for extraction of specific results, which the cloud-side analytic framework then passes back to its client-side counterpart, and from there on to the analysis nodes in the authoring tool GC. In GC they are now available to the user for further computations.

The optimization workflow adds the Design Evolution (DE) node, recently renamed and in the following called Optimizer node, which is the authoring representation of the optimization framework. In the prototype the optimization is the DARWIN optimization framework which includes a prototype implementation of a multi-objective genetic algorithm (MOGA) [4]. In the Optimizer node users specify the parameters for the MOGA run, like population size (i.e. how many individual design solutions per generation), how many generations, termination criteria, etc. Users wire into the Optimizer node (a) the parameters or design variables that they want to be controlled by the optimization framework and (b) the decision values or fitness functions based on which the optimization framework (or the embedded MOGA, respectively) makes decisions about the next generation of individuals. Decision values can either be analysis results directly wired into the Optimizer node or any other computed values that are dependent on any of the design variables. Once the analysis nodes are set to execute whenever they receive a propagated change from upstream, the optimization run can be started by the user.

4. Prototype limitations

The prototype implementation had severe limitations:

(1) Because only the analysis engines resided in the cloud, there was a three-fold penalty for each analysis: time spent uploading the analytical model data to the cloud, lower performing standard cloud compute nodes taking longer for the analysis computation proper, and additional time spent downloading the results from the cloud to the client.

(2) The communication between the prototype Optimizer node and the prototype optimization framework used a socket connection implemented in a way that minor disturbances caused total disruption.

(3) Because the prototype optimization framework used the parametric model in the same GC instance from which the process was started, all individuals in each generation had to be processed sequentially. Parallel processing of all individuals in one generation would have been the greatest benefit of a cloud implementation; however, this was not achieved with the prototype.

(4) In addition, for the optimization use case the cloud penalty was multiplied because it applied to each individual in each generation.

(5) This means that the only parallelization was potentially concurrent analysis runs for any single individual.

(6) The energy analysis model did not include shading elements.

(7) One assumption in the structural model was that the lowest structural nodes are the nodes connecting to the foundation. While this may be a correct assumption for many buildings, it

creates problems with buildings on sloping sites, or structures that are supported in turn by other structures.
(8) In general, there was a lack of robustness.

5. Lessons learned for second implementation iteration

In order to gain full benefit of cloud resources for the optimization use case, it is necessary to minimize the effect of cloud penalties. This means that ideally there is only one upload to the cloud and one download from the cloud, and as much parallel processing in the cloud as possible. Figure 2 shows the corresponding changes in software architecture. Given the complexity of the software architecture, the multitude of systems and services involved, and the complexity of process flows, it became clear that the modules involved in that architecture need to be sufficiently robust. This means that instead of stringing together multiple prototype implementations with shortcomings and points of failure decreasing stability almost exponentially, modules need to have production-strength qualities.

Consequently, S2I comprises modules from industry-proven solutions and modules re-written to meet production-level standards. The authoring environment (GC), most of the node types in it, the analytical framework, and the analysis engines had either been written to meet these standards, or were already available as production-level modules, respectively. The optimization engine was replaced by a production-level engine which has been in use for several years. This optimization engine was extended to become a true multi-objective engine. The optimization framework is still being re-written, as well as the DE node. Several of the necessary cloud services have been or will be swapped out against production level modules which will be provided by a different R&D group.

Optimization in the cloud is still in development at the writing of this paper. The S2I parallelizes computation of individuals per generation. Goal is also that any analysis can be triggered in a cascading fashion. For a generation with a population with 100 individuals and utilization of 2 analyses per individual, there are potentially 300 processes that could be executed in parallel. Because the MOGA still needs to wait until all results for a generation are returned as decision values, the critical timeline is the slowest combination of model generation and analysis for each generation.

Other opportunities afforded by this system have already been implemented in S2I to utilize the work and spread the benefit of having the developed cloud resources at hand. All use cases as implemented already or in the near future are illustrated in Figure 3:
(1a) With the STAAD structural analysis engine available in the cloud, incremental effort allowed the STAAD. Pro structural design application off-loading of analysis runs to the cloud. These can be run sequentially and later inspected through a web-based scenario manager application to explore how the design's performance changed during design development.
(1b) Alternatively, and using the virtually unlimited availability of cloud resources, a designer can set up multiple design scenarios in STAAD. Pro, and then send them off to the cloud concurrently, thus parallelizing the analyses and shortening wait time for the results.

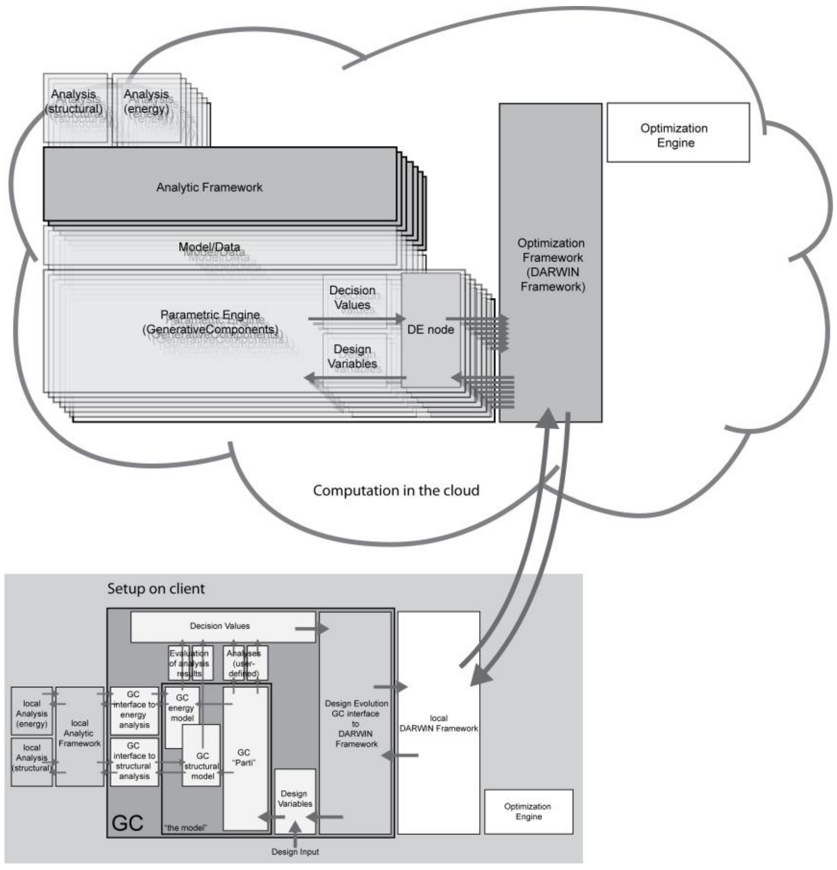

Figure 2. Software architecture for S2I's optimization use case.

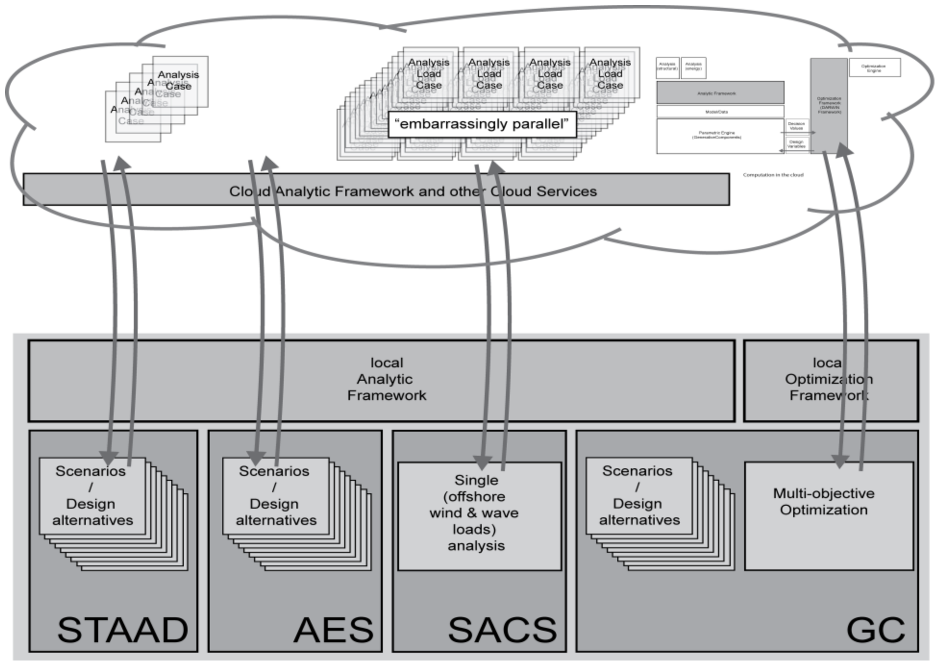
Analysis Case
Analysis Load Case
"embarrassingly parallel"
Cloud Analytic Framework and other Cloud Services
Computation in the cloud
local Analytic Framework
local Optimization Framework
Scenarios / Design alternatives
Single (offshore wind & wave loads) analysis
Multi-objective Optimization
STAAD
AES
SACS
GC

Figure 3. S2I multi-client and cloud architecture with three distinct use cases.

(2) The SACS design and analysis application for offshore structures generates many load cases for wave and wind loads for a single analysis. These load cases are independent of each other. Customarily, SACS analysis runs on a desktop can last hours, days, or even weeks. That is unfeasible for design iterations. With hundreds of independent load cases that can be analyzed in parallel, this type of "embarrassingly parallel" process is an almost classic use case for high performance computing.

(3) The cloud-based optimization use case, shown in detail in Figure 2.

Further improvements have been:

- Amendment of the structural model with surface elements for controlled application of wind loads.
- Addition of a wind load node type that allows definition of custom wind loads for regions for which no wind load data are available.
- Tagging of structural nodes when they are connecting to supports, e.g. foundations, which avoids problems otherwise caused by non-standard situations that do not meet the default expectations.
- Addition to the energy model of a shading node type extending the range of conceptual energy explorations.
- Addition of chart types to enable data visualizations in the design authoring environment GC.
- Enhancement of the color system in GC for enhanced analysis result visualization in the geometric model.
- Addition of a simpler analysis type indicative of energy influx.

6. First benchmark

With the SACS analysis use case as the most compelling use case implemented in this R&D work, it is of interest to validate use of cloud resources. Benchmark case was the fatigue analysis for a connection between the base of an offshore wind turbine tower and the tower itself (Figure 4).

The benchmark was conducted on a multi-core desktop, using a controlled number of cores, and for comparison using the cloud implementation utilizing the analysis framework and related cloud services, also with set numbers of compute nodes dedicated to this analysis. The benchmark results are shown in Table 1.

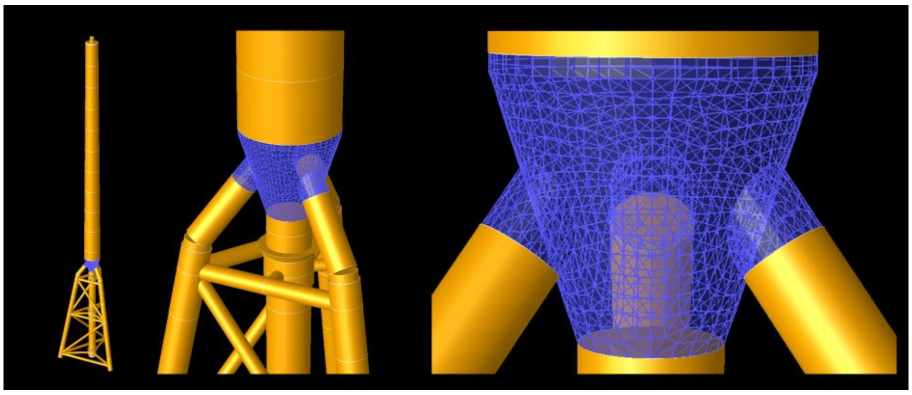

Figure 4. The part of an offshore wind turbine tower that were analyzed for the benchmark.

As can be seen, the processing time is in inverse dependency of the number of cores and nodes (Figure 5). The cloud penalty can be seen in the dual-core / two node comparison: the raw processing time is slower in the cloud because a safe assumption is that a node in the cloud has lower performance than a core in a higher-end mobile workstation. In addition there is the overhead time of uploading the analysis case to the cloud and downloading the results back to the client machine. This overhead is in fact constant for the same case (i.e. the same amount of data needs to be uploaded and downloaded), while the processing time decreases consistent with the number of cores or nodes, respectively. Towards the high number of cloud compute nodes, the overhead starts to diminish the return of increased resources significantly. In the last test case, the processing time was already less than the overhead time. Considering per-node costs, further increase of node numbers would not make sense. Perhaps the reasonable cut-off would occur even earlier.

TABLE 1. SACS "embarrassingly parallel" analysis use case.

Number of Cores	Total Time		Number of Nodes	Processing Time	Overhead Time	Total Time
1	8,100.0					
2	4,275.0		2	5,850.0	40.0	5890.0
4	2,193.0					
8	1,293.0					
			20	607.5	37.0	644.5
			40	297.0	37.5	334.5
			80	140.4	38.0	178.4
			150	77.4	39.0	116.4
			300	37.5	39.0	76.5

7. Remaining work

Work in progress is the completion of the Optimizer node and the optimization framework. Once this work is complete, benchmarking of local optimization runs versus cloud-based optimization runs would validate the assumptions about time savings. Depending on the types of analyses involved, it is yet another step to determine feasibility of the approach through as early an involvement of users as is possible, once basic functionality has been established. User involvement will help determine whether assumptions about the analysis model node types have been sufficient, and how defaults may need to be adjusted or exposed for user intervention. Addition of other analysis types and optimization algorithms are other options available in the framework.

8. Conclusions

Although the vagaries of designing, developing, and writing software are well-known, a research and development project like the one described shows that innovative development has additional challenges, the infamous "unknown unknowns" that need to be resolved in order to get everything to work as designed. Especially for the optimization use case all modules need to be in place and need to function robustly and reliably. Therefore, it was significant that in the preparation of the second phase of this research related workflows became apparent which utilize parts of the infrastructure necessary for the optimization use case.

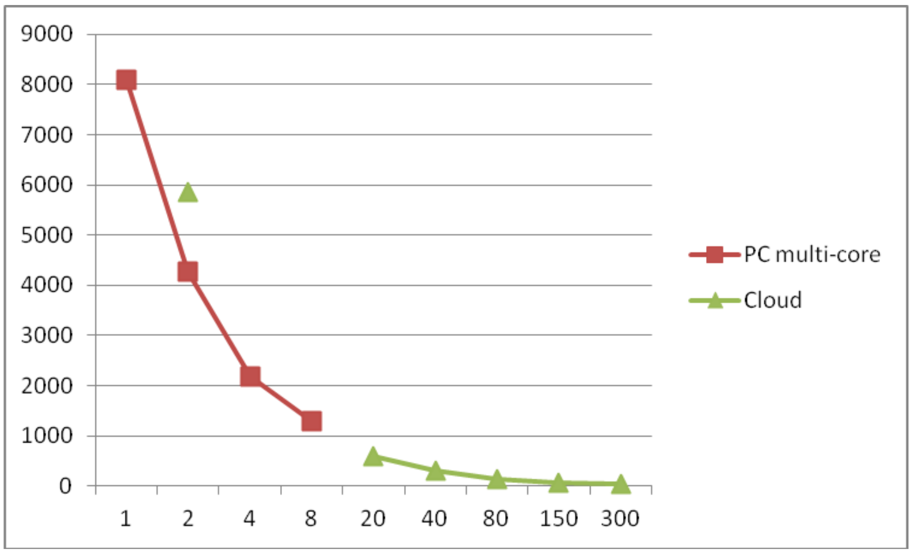

Figure 5. (Almost) Logarithmic plot of total processing time in seconds against number of cores and nodes.

With this scope change in the S2I from focusing on the parametrically driven multi-objective optimization case to extending the R&D work to include access of the analysis framework from other design authoring applications and addition of other analysis engines in the cloud the working title of the paper "design performance optimization framework" becomes a misnomer. Given the potential impact and actual intent, the operational term is proposed to become "design performance improvement framework" which is accurate for all use cases.

The shared goal for all use cases is to learn about any design variant as much as possible as quickly as possible, and to be able to compare variants' performance characteristics to understand how design changes affect the behaviour of the design variants. The deeper purpose is, of course, to support designers to create better designs.

Acknowledgements

This work was supported by contributions from various groups within Bentley Systems, Incorporated, especially the Applied Research Group, the Special Technology Projects group, Design and Simulation's Structural Analysis team, and the Bentley CONNECT team.

Endnotes

1. GenerativeComponents from Bentley Systems: http://www.bentley.com/en-US/Products/GenerativeComponents/. Accessed May 24, 2013.
2. STAAD.Pro user interface to the STAAD analysis engine: http://www.bentley.com/en-US/Products/STAAD.Pro/. Accessed May 23, 2013.

3. EnergyPlus from the U.S. Department of Energy, available at http://apps1.eere.energy.gov/buildings/energyplus/. Accessed May 23, 2013.
4. Darwin Optimization (version 0.91) by Dr. Zheng Yi Wu. Available from: Be | Communities http://communities.bentley.com/communities/other_communities/bentley_applied_research/w/bentley_applied_research__wiki/6584.aspx. Accessed Dec. 7, 2012.

References

BLEIL DE SOUZA, C., 2012. Contrasting paradigms of design thinking: The building thermal simulation tool user vs. the building designer. *Automation in Construction, Volume 22*, 112–122.

DONDETI, K. AND REINHART, C.F. 2011. A "Picasa" for BPS – An interactive data organization and visualization system for building performance simulations. *In:* SOEBARTO, V. AND WILLIAMSON, T. (eds.) *Proceedings of Building Simulation 2011: 12th Conference of International Building Simulation Association*, Sydney, Australia.

FLAGER, F., SOREMEKUN, G., WELLE, B., HAYMAKER, J. AND BANSAL, P.: 2008. Multidisciplinary process integration and design optimization of a classroom building, *CIFE Technical Report TR175*, Stanford University.

HETHERINGTON, R., LANEY, R., PEAKE, S. AND OLDHAM, D., 2011. Integrated building design, information and simulation modelling: the need for a new hierarchy. *In: Proceedings of Building Simulation 2011*, Sydney, 2241-2248.

JANSSEN, P., STOUFFS, R., CHASZAR, A., BOEYKENS, S. AND TOTH, B.: 2012. Data transformations in custom digital workflows: Property graphs as a data model for user-defined mappings. *In: International Workshop: Intelligent Computing in Engineering 2012 - EG-ICE 2012*, Munich, Germany.

MUELLER, V., CRAWLEY, D.B., AND ZHOU, X.: 2013. Prototype implementation of a loosely coupled design performance optimisation framework. In: *Open Systems: Proceedings of the 18th International Conference of the Association of Computer-Aided Design Research in Asia*, The Association for Computer-Aided Architectural Design Research in Asia (CAADRIA), Singapore, 675-684.

TOTH, B., BOEYKENS, S., CHASZAR, A., JANSSEN, P. AND STOUFFS, R.: 2012. Custom digital workflows: A new framework for design analysis integration. *In:* FISCHER, T., DE BISWAS, K., HAM, J.J., NAKA, R. AND HUANG, W.X. (eds.) *Beyond Codes and Pixels: Proceedings of the 17th International Conference on Computer-Aided Architectural Design Research in Asia*, The Association for Computer-Aided Architectural Design Research in Asia (CAADRIA), Hong Kong, 163–172.