



GREEN IT

Modeling the impact of software development

Michel Dejolier

Table of Contents

Green IT	2
Why capacity planning?	2
Maturity of processes	5
Green software (SCI)	6
Basic capacity planning	14

Green IT

IT can represent a large part of energy consumption, water usage and waste management of an enterprise and as such, deserves a dedicated model. This document describes a fully functional prototype that implements the recommendations of the Green Software Foundation and adds some capacity planning and modeling capabilities. It is part of a broader toll that also implements all recommendations of the GREEN GRID to calculate the data center KPIs like PUE, WUE, DCcE, etc. and has been primarily developed to support a new “IT sustainability” curriculum I am developing for the University of Singapore.

Although it is not developed as plugins to your IF framework, simply because I was working on this long before your contest was announced, each function is independent and could be transformed into a plugin. To give you an idea, my first version of Green IT was operational in 2011 and I used it as a consultancy tool with some customers, but I never wanted to make it a commercial product: I have been in this business in the 90's when my capacity planning tool was acquired by an American company and distributed worldwide, so I can tell it is just exhausting to be a lonely developer of a commercial product and the priority today should be having a real impact on climate change instead of making more money that will be useless on a dead planet.

The platform capabilities of the Green Software section are:

- Assessing the maturity of Green Software processes in the company
- Measure an existing application, a part of it, or an application under development
- Model changes to the software
- Use basic capacity planning functions to evaluate the impact of a change on CPU queue

Why capacity planning?

Capacity planning could be used at two different levels in the Green IT methodology to better serve the governance goals:

First, to determine the capacity that is actually required to run an application, so that server virtualization can be used in an effective way. People think that virtualization is THE solution to reduce carbon emissions, but it is very common to see ineffective virtualizations that make the problem even worse.

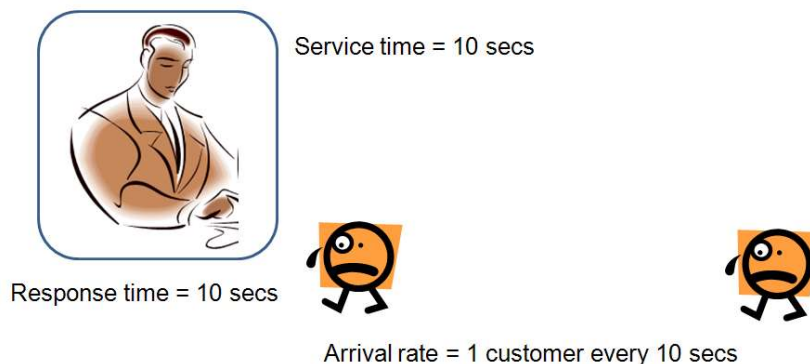
Next, capacity planning should be used when designing a new project so that only the required resources are added to the current environment, or even to allow available resources to be used, thereby improving DCcE automatically.

There is no need for a sophisticated capacity planning tool such as the one we used for mainframes in the 90's. When it is about Green IT, the tool must just be able to evaluate the CPU power and DASD requirements needed to provide the target response time, and model the

effects of workload increase over time so that additional capacity can be installed when needed.

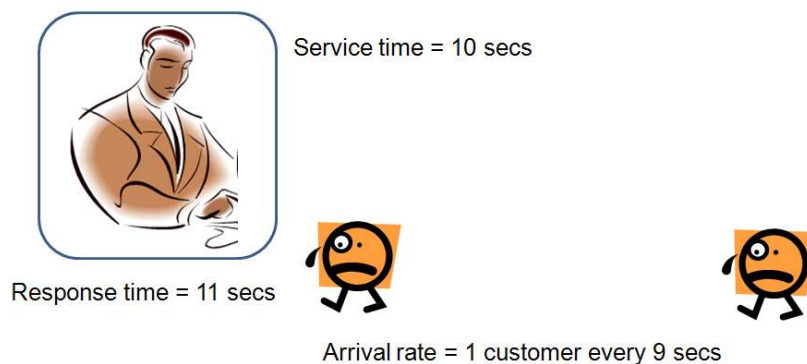
The point is, if you use capacity planning, then you are able to model the impact of a new application on your green KPIs well before it is ready to move into production. If you don't use capacity planning, you can still model the impact of the new application on the environment, although in a less accurate way, by estimating the resource requirements, both at CPU level and at DASD level.

So, let's have a look at the basics of predictive modeling with a very simple example applied to capacity planning in the broad sense, that is, not limited to computer networks but also applicable to any activity that is subject to waiting lines, like the processing of a service request, the access to a support center or the processing of a retirement application. At the main entrance of a theater, there is a ticket window where the employee needs 10 seconds to process a request. The interval between the arrival of two customers is also 10 seconds:



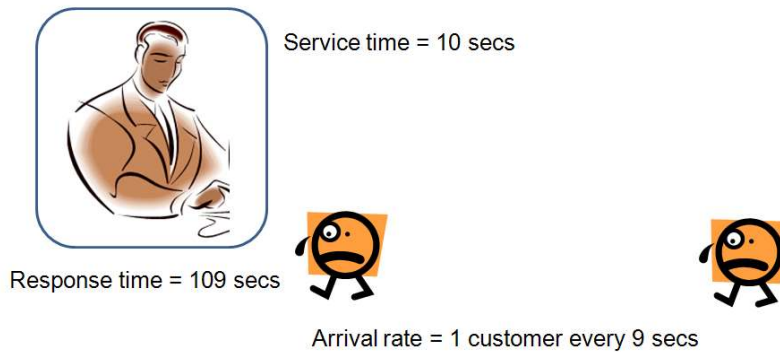
In these conditions, the response time is stable at 10 seconds. No waiting time at all.

Imagine that the arrival rate increases slightly, and we now have a customer every 9 seconds:



At the beginning, which is for the first customer, there is no change; the response time is still 10 seconds. For the second customer, the response time goes up to 11 seconds, nothing dramatic. But for the 100th customer, things are very different:

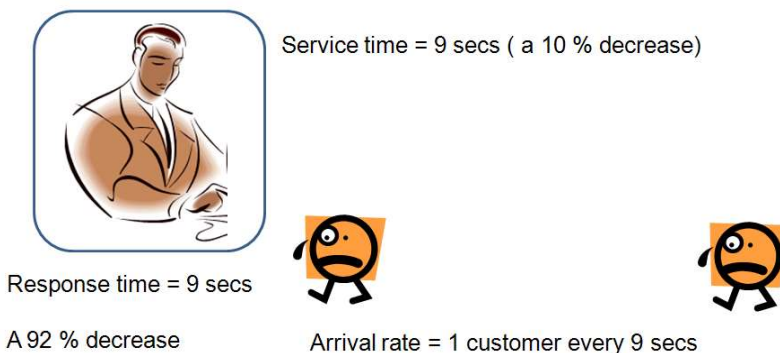
For customer # 100



The response time has now gone up by more than 1000 % and will theoretically go up indefinitely. In real life, things will stabilize at a very bad level, because customers will simply give up at some point in time.

Predictive modeling can calculate what the **average** queue length will be in these circumstances, without the need to simulate the employee and the customers. The model can either be a quite simple $M/M/1$ formula or a very complex set of formulas that take into account the fact that customers will not arrive at a constant rate of 1 every 9 seconds, but rather in bursts. If it is a simple $M/M/1$, you can probably do it yourself. If it is a complex model, you can't, unless you have a team of experts like some industries have because it is critical to their business. Think about NASA or Boeing...

We have just described a predictive model that will tell you how bad the situation could become in the future if you don't take preventive decisions. Another model could be used to evaluate the effectiveness of your decision. Guess what happens if we decrease the "Service time" of the employee to 9 seconds:



We will come back to this later.

Maturity of processes

Evaluating the maturity of Green Software processes should be the first step for a weakness to be identified and corrected. It is based on a set of KPIs that request a score between 0 and 5 in most cases, or a Yes/No answer.

Each KPI is associated to an icon and the global score is calculated as the average of all answers, defining the color of the top item.

The function can be used regularly to confirm that the situation has been improved.

Here is an example:

Green Software processes assessment

Select an assessment: Michel - Nov 23 2023 12:00AM Create new assessment


- Green Software Score
- Optimizing Code**
- Energy-Aware Design Patterns
- Cloud Computing
- Green Software Development Tools
- Dynamic Resource Allocation
- Energy-Efficient Algorithms
- Eco-Friendly Software Deployment
- Promoting Green Practices
- Lifecycle Assessment
- Monitoring and Optimization
- Server Utilization
- Calculation of SCI

Writing efficient code that minimizes resource consumption. Techniques like code refactoring, using algorithms with lower computational complexity, and optimizing database queries can significantly reduce energy consumption.

Maturity: ☐ 0 (No)
☐ 1
☐ 2
☐ 3
☒ 4
☐ 5 (Yes)

Binary (Y/N): ☐ 0 (No)
☒ 1 (Yes)
☐ 2 ??

Score



Close session

Green software (SCI)

The Green software development model based on the work of the Green Software Foundation is actually capacity planning that doesn't say its name because the main goal of this model is to decrease the resources usage both during the development cycle and during production.

Software systems cause emissions through the hardware that they operate on, both through the energy that the physical hardware consumes and the emissions associated with manufacturing the hardware. Many computer manufacturers now deliver a data sheet specifying the carbon footprint of manufacturing and decommissioning the hardware at the end of its life. They also include the energy consumption over an estimated lifespan of 4 years, but this is useless as it's much better to measure the actual resources usage and use the lifespan that you estimate for this particular server to derive the embedded emissions allocated to a task.

I have used the acronym SCI in my modeling tool and it refers to the amount of Co2 emitted by an application or a part of an application that is being monitored. Reducing an SCI score is only possible through the elimination of emissions. That can be achieved by modifying an application to use less physical hardware and therefore less energy.

I have used the symbols defined by the Green Software Foundation throughout the methodology:

- E - Energy consumed by a software system
- I - Location-based marginal carbon intensity
- M - Embodied emissions of the hardware needed to operate a software system
- O - Operational emissions based on the emissions caused by energy consumption
- R - Functional unit

So, a simple definition of SCI is $SCI = C \text{ per } R$

or $SCI = (O + M) \text{ per } R$

Embodied carbon (otherwise referred to as “embedded carbon”) is the amount of carbon emitted during the creation and disposal of a hardware device.

When software runs on a device, a fraction of the total embodied emissions of the device is allocated to the software. This is the value of `M` that needs to be calculated in the SCI equation.

This fraction consists of both a time- and resource-share. The length of time that the software runs on the device determines its time-share. The percentage of the device reserved just for that application during the time-share determines that application's resource-share.

$$M = TE * \left(\frac{TIR}{EL} \right) * (RR/TOR)$$

Where

TE=Total embodied emissions , which is basically the Co2 footprint of building the server and disposing of it at end of life

TIR=Time reserved: the amount of time that the software is using the server

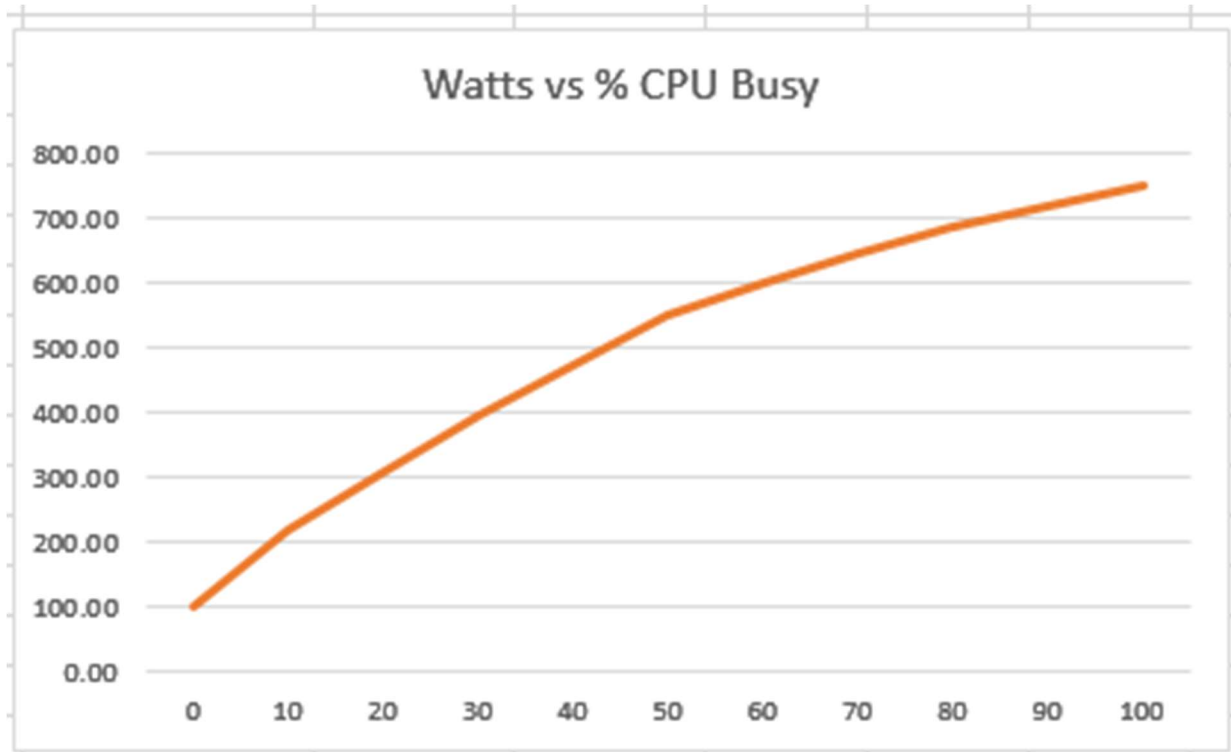
EL: expected lifespan of the equipment. Not that elongating the lifespan has a positive effect on Co2

RR: Resources reserved for the software: it can be CPU time, disk I/O time, etc.

TOR: Total resources available. In my opinion, more tricky because even though 100 % would be the best for Co2 emissions it would most probably cause a severe performance degradation.

So, in the model I developed for this tool, I elected to use only 3 types of resources RR to simplify the use: CPU, I/O rate and MB transferred per second. Lets' take a practical example, it will make things much easier to understand.

One more thing: the power usage of a CPU is not directly proportional to the percentage CPU busy and it would be incorrect to say that if a server has a max power usage of 750 watts, it is using 7.5 watts per 1% CPU . So, I have used the data of multiple servers to derive an approximation of the energy curve with a standard formula that can be applied to any server with an acceptable accuracy. If the user has a better data set about power usage vs resource usage, he can modify the formula accordingly. Here is the standard curve:



It shows that the server uses 100 watts when idle and that the inflexion point is around 50 % CPU busy. Above that value, the curve tends to flatten and the emissions avoided by reducing power usage become smaller.

One could say “Let’s limit the CPU usage to 50 %”, but that’s not such a good idea for obvious reasons although in real life, I have seen a lot of servers used at much less than 50 %, basically for performance reasons.

This is an example for an existing software that we will try to optimize in the What-If section of the tool to reduce both emissions and cost.

Open software model: Software test (06-Mar-24) [Create new software] [Save software] [Export to Excel] [Rename software] [Delete software]

0.38

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	Co2 footprint manufacturing	1230 kg																
2	Lifespan	4 years																
3	Power usage/hour	750 watts																
4	Power usage per % cpu	3.98 watts																
5	Co2 Coeff	0.38 kg/kWh																
6	% cpu busy	80 %		Cpu Usage														
7	Io rate/sec	50 Number		2 %														
8	MB/Sec	200 Number		1.5 %														
9																		
10	My software uses:				Power		Usage footprint		Embodied		Total							
11		15 % cpu			477 watts		181.26 grams/day		47.13 Kg/year		46.13 Kg/year		93.25 Kg/year					
12		12 I/O / sec			477 watts		181.26 grams/day		47.13 Kg/year		46.13 Kg/year		93.25 Kg/year					
13		20 MB/sec			477 watts		181.26 grams/day		47.13 Kg/year		46.13 Kg/year		93.25 Kg/year					
14													93.25 Kg/year		Average			
15				CPU % per unit														
16	Runs	8 hours/day											SCI					
17		50 users		2.40	9.54 watts		3.63 grams/day		0.94 Kg/year		0.92 Kg/year		1.87 Kg/User					
18		1200 Transactions		0.10	0.40 watts		0.15 grams/day		0.04 Kg/year		0.04 Kg/year		0.08 Kg/Trx					
19		5 Days/week																
20																		
21																		
22																		

So, in the green cells are all the parameters that affect the result of the model, and here we are just in the definition part and we enter the real data as measured. The Co2 footprint of manufacturing the server is 1230 kg and this information is given by the vendor. For example, DELL gives a detailed data sheet with each server.

The lifespan has been set at 4 years and the nominal power usage per hour at 750 watts/h. So, using the curve described above, the watt usage per CPU % is calculated.

The Co2 coefficient is country dependent and even sometimes location dependent in large countries like USA.

% CPU busy, I/O rate and MB/sec are all coming from measurement and this concludes the general part of the model.

Then, we enter the parameters of the software we are scrutinizing, in this example 15 % CPU, 12 I/O per second and 20 MB/sec.

We also specify how many hours per day the software is running, how many users connected (average) and number of transactions during these 8 hours (if relevant), and how many days per week the software is in use.

These 3 lines 11,12 and 13 where you enter CPU, I/O and Transfer rate give the same calculation values in terms of CPU power used and Co2 emissions, so they could look duplicate but they are not as they will be used as reference during the what-if sessions.

And for each line, the amount of Co2 generated by usage is calculated, the portion of embodied Co2 footprint is calculated and this gives a total of Co2 Kg based on CPU usage.

We also have a footprint per user and a footprint per transaction. And finally, the SCI calculated both per user and per transaction and this is the sum of embodied emissions and usage emissions.

Now that you have an idea of what your software is using in terms of resources, you can decide what you will try to model first. I am certainly not saying that you should model several changes in the same What-if model unless you have a lot of experience in modeling, because the results could get quite confusing.

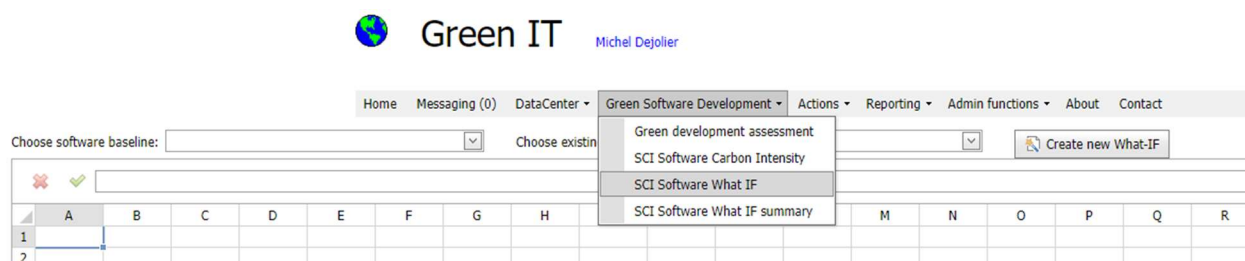
If you plan to model changes at the I/O rate or MB transfer rate level, you must supply the percentage of the CPU that is used by these function. This can be found in specific performance reports depending on the kind of file system or database you are using. In columns D7 and D8, you enter the CPU usage corresponding to 50 I/O per second and 200 MB/sec.

As I said, I only model based on user and transaction, but more model templates could be added in the future is a customer expresses interest for them, mainly:

- API call/request
- Benchmark
- Machine
- Time unit
- Transaction
- Database read/write

I limited the model to two units of work as it is already quite complex and it would be best to have separate templates for other unit types.

So, let's take an example of modeling, menu Green software development, Green software what-if:



The first thing to do is to select a baseline model containing the latest measurement data. Then chose an existing what-if or create a new session:

Choose software baseline: Choose existing What-IF:

I select to create a new one:

What-IF name:*

This creates a new What-if session and at first the data from the measurement and the what-if data are the same. The goal will be to modify some parameters to evaluate the effect on both resources usage and Co2 emissions.

Choose software baseline:

Choose existing What-IF:

	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	1230 kg				What If	Co2 footprint manufacturing		1230 kg											
2	4 years					Lifespan		4 years											
3	750 watts					Power usage/hour		750 watts											
4	3.98 watts					Power usage per % cpu		3.98 watts											
5	0.38					Co2 Coeff		0.38											
6	80 %		Cpu Usage			% cpu busy		80 %											
7	50 Number		2 %			Io rate/sec		50 Number											
8	200 Number		1.5 %			MB/Sec		200 Number											
9																			
10				Power			Usage footprint			Embodied			Total						
11	15 % cpu				477 watts		181.26 grams/day		47.13 Kg/year		46.13 Kg/year		93.25 Kg/year						
12	12 I/O / sec				477 watts		181.26 grams/day		47.13 Kg/year		46.13 Kg/year		93.25 Kg/year						
13	20 MB/sec				477 watts		181.26 grams/day		47.13 Kg/year		46.13 Kg/year		93.25 Kg/year						
14													93.25 Kg/year		Average				
15			CPU % per unit																
16	8 hours/day												SCI						
17	50 users		2.40	9.54 watts		3.63 grams/day		0.94 Kg/year		0.92 Kg/year			1.87 Kg/User						
18	1200 Transactions		0.10	0.40 watts		0.15 grams/day		0.04 Kg/year		0.04 Kg/year			0.08 Kg/Trx						
19	5 Days/week																		
20																			
21																			
22				Power			Usage footprint			Embodied			Total					Saved	
23	15 % cpu				477 watts		181.26 grams/day		47.13 Kg/year		46.13 Kg/year		93.25 Kg/year						0.00
24	12 I/O / sec				477 watts		181.26 grams/day		47.13 Kg/year		46.13 Kg/year		93.25 Kg/year						0.00
25	20 MB/sec				477 watts		181.26 grams/day		47.13 Kg/year		46.13 Kg/year		93.25 Kg/year						0.00
26													93.25 Kg/year			SCI TOTAL			0.00
27			CPU % per unit																
28	8 hours/day												SCI						
29	50 users		2.40	9.54 watts		3.63 grams/day		0.94 Kg/year		0.92 Kg/year			1.87 Kg/User						
30	1200 Transactions		0.10	0.40 watts		0.15 grams/day		0.04 Kg/year		0.04 Kg/year			0.08 Kg/Trx						
31	5 Days/week																		

In this session, we will set a goal of CPU usage at 12 % if we think that this is a reasonable target that can be attained by optimizing code. And here is the result:

	Power		Usage footprint		Embodied	Total		Saved
12% cpu		382 watts	145.01 grams/day	37.70 Kg/year	36.90 Kg/year	74.60 Kg/year		18.65
12 I/O / sec		382 watts	145.01 grams/day	37.70 Kg/year	36.90 Kg/year	74.60 Kg/year		0.00
20 MB/sec		382 watts	145.01 grams/day	37.70 Kg/year	36.90 Kg/year	74.60 Kg/year		0.00
						74.60 Kg/year	SCI TOTAL	18.65
	CPU % per unit							
8 hours/day						SCI		
50 users	1.92	7.63 watts	2.90 grams/day	0.75 Kg/year	0.74 Kg/year	1.49 Kg/User		
1200 Transactions	0.08	0.32 watts	0.12 grams/day	0.03 Kg/year	0.03 Kg/year	0.06 Kg/Trx		
5 Days/week								

The total Co2 footprint of this application would fall from 93.25 kg to 74.6 Kg. The SCI per user would fall from 1.87 to 1.49 and the SCI per transaction from 0.08 to 0.06.

We can create as many different models as we want and the results will all appear in the reporting sub-menu of SCI modeling, where we can see all what-if models that have been created for a specific baseline and we can also export them to excel.



Green IT

Michel Dejolier

[Home](#)
[Messaging \(0\)](#)
[DataCenter ▾](#)
[Green Software Development ▾](#)
[Actions ▾](#)
[Reporting ▾](#)
[Adr](#)

Green development assessment
SCI Software Carbon Intensity
SCI Software What IF
SCI Software What IF summary

GreenIT what-if summary

Software name	AVG CO2 Kg per year		CPU CO2 Kg per year		I/O CO2 Kg per year		MB/s CO2 Kg per year		SCI per User	SCI per Transaction	export
Software test {06-Mar-24}	93.25		93.25		93.25		93.25		1.87	0.08	
What-if name	AVG Co2 current	AVG Co2 Projected	Co2 CPU before	Co2 CPU after	Co2 I/O before	Co2 I/O after	Co2 MB/s before	Co2 MB/s after	SCI per User	SCI per Transaction	
Software test from lab 3	93.25	74.60	93.25	74.60	93.25	74.60	93.25	74.60	1.49	0.06	
Software test what if 1	93.25	80.82	93.25	80.82	93.25	80.82	93.25	80.82	1.62	0.07	
Software test from lab	93.25	79.64	93.25	80.82	93.25	80.82	93.25	79.64	1.60	0.07	

In real life, the list can be very long as there will be one main grid line for each evaluated software and n lines that can be opened individually for each software, showing the details of the various models.

If you are a beginner in modeling, it's always a good idea to do it one step at a time, but as you gain experience, you might want to model more than one change at a time, but the results must then be analyzed properly as they could get a little confusing.

Let's take back the same model, but now we will evaluate both a CPU usage change and an I/O rate change:

	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	1230 kg				What if	co2 footprint manufacturing		1230 kg									58		
2	4 years					Lifespan		4 years											
3	750 watts					Power usage/hour		750 watts											
4	3.98 watts					Power usage per % cpu		3.98 watts											
5	0.38					Co2 Coeff		0.38											
6	80 %	Cpu Usage				% cpu busy		80 %											
7	50 Number		2 %			Io rate/sec		50 Number											
8	200 Number		1.5 %			MB/Sec		200 Number											
9																			
10						Power		Usage footprint					Embodied			Total			
11	15 % cpu			477 watts			181.26 grams/day		47.13 Kg/year			46.13 Kg/year				93.25 Kg/year			
12	12 i/O / sec			477 watts			181.26 grams/day		47.13 Kg/year			46.13 Kg/year				93.25 Kg/year			
13	20 MB/sec			477 watts			181.26 grams/day		47.13 Kg/year			46.13 Kg/year				93.25 Kg/year			
14																93.25 Kg/year	Average		
15					CPU % per unit														
16	8 hours/day															SCI			
17	50 users		2.40	9.54 watts			3.63 grams/day		0.94 Kg/year			0.92 Kg/year				1.87 Kg/User			
18	1200 Transactions		0.10	0.40 watts			0.15 grams/day		0.04 Kg/year			0.04 Kg/year				0.08 Kg/Trx			
19	5 Days/week																		
20																			
21																			
22						Power		Usage footprint					Embodied			Total			Saved
23	12 % cpu			382 watts			145.01 grams/day		37.70 Kg/year			36.90 Kg/year				74.60 Kg/year			18.65
24	10 i/O / sec			371 watts			140.98 grams/day		36.85 Kg/year			36.90 Kg/year				73.55 Kg/year			1.05
25	20 MB/sec			371 watts			140.98 grams/day		36.65 Kg/year			36.90 Kg/year				73.55 Kg/year			0.00
26																73.55 Kg/year	SCI TOTAL		19.70
27					CPU % per unit														
28	8 hours/day															SCI			
29	50 users		1.89	7.52 watts			2.86 grams/day		0.74 Kg/year			0.74 Kg/year				1.48 Kg/User			
30	1200 Transactions		0.08	0.31 watts			0.12 grams/day		0.03 Kg/year			0.03 Kg/year				0.06 Kg/Trx			
31	5 Days/week																		

We bring down the I/O rate from 12 to 10 per second on top of the CPU reduction already recorded. We can see that we save an additional 11 watts, mainly because of the CPU cycles saved by reducing I/O. The disk power usage does not change.

Here is the tricky part (just a little):

With CPU change, we have saved $93.25 - 74.6$ kg of co2 = 18.65 kg

With I/O change, we have saved an additional 1.05 kg, which brings the total emissions down to 73.55 kg and the SCI per user to 1.48.

Another model that would prove useful in software carbon intensity is the one dealing with software developments that are canceled (around 35 % of all developments), software that don't deliver all expected features and yet consume a lot of resources.

This model is available under the menu "Data models" and the name is "Canceled projects". It allows to evaluate how much energy could be saved and how much Co2 emissions could be avoided if the planning phase was better executed.

Here is an example:

Facts						
31% of US IT projects are canceled outright and the performance of 53% is so worrying that they are challenged		Research from McKinsey in 2020 found that 17% of large IT projects go so badly, they threaten the very existence of the company.				
How many projects were canceled:	5	Energy lost server	840,000 Watts			
What % of total projects:	21%	Energy lost Laptops/desktops	100,800 Watts			
How many days Lost:	140	Total energy lost	940,800 Watts			
How many Laptops/desktops used:	3	Co2 emissions	366,912 Kg		366.91 Tonnes	
How many servers involved:	1					
Watts per server per hour:	750					
Co2 factor electricity	0.39					

All green data fields can be modified to reflect your situation and it takes into account both the development / Test servers and the laptops and desktops involved in the failed developments.

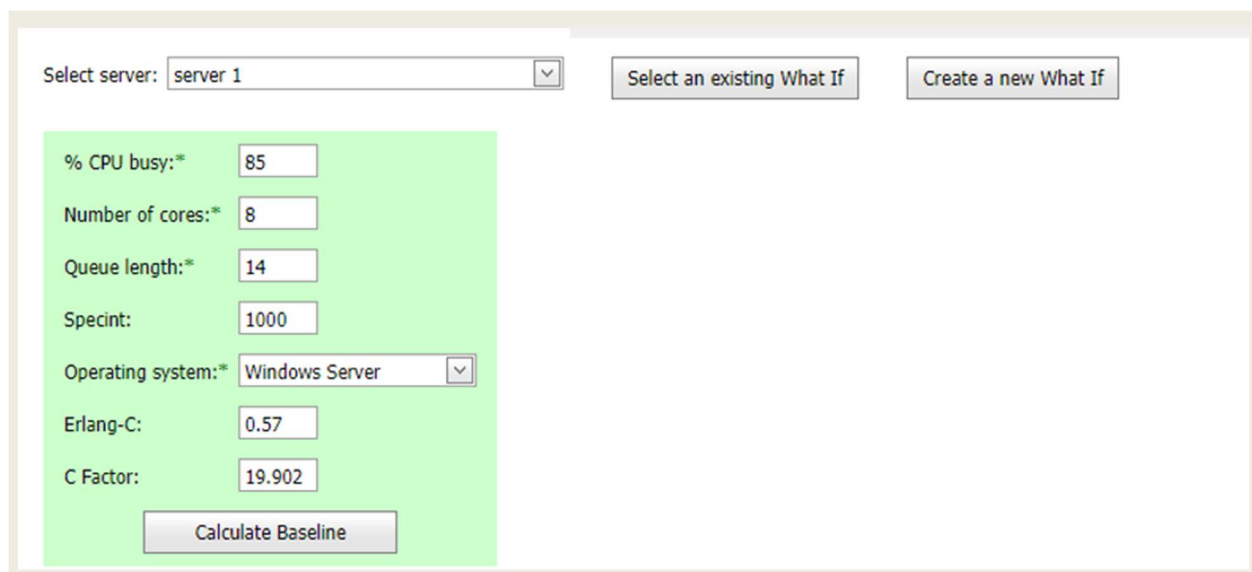
Basic capacity planning

We have talked before about the basic queueing theory and the exponential effect of any component becoming overused in a computer system.

A green IT tool without any capacity planning may leave people wondering if their decision to virtualize several machine is a good one, if a simulation they made on the Software Carbon Intensity (SCI) will have a significant effect on performance on top of reducing emissions, and so on. Therefore, we have included a basic model to evaluate CPU performance under various conditions.

It's called Basic because it only covers CPU, not disk drives, not network, not memory, not the response time at transaction level because this would be totally beyond the scope of Green IT, but it nevertheless uses the state of the art algorithms for predicting Queue length at the processor level.

Any server that has been enter in the data center efficiency page can be analyzed using this model. Just select the server in the “Basic Capacity Planning” page:



The screenshot shows a web interface for "Basic Capacity Planning". At the top, there is a "Select server:" dropdown menu with "server 1" selected. To the right of this are two buttons: "Select an existing What If" and "Create a new What If". Below these is a green-bordered box containing several input fields: "% CPU busy:*" with the value 85, "Number of cores:*" with the value 8, "Queue length:*" with the value 14, "Specint:" with the value 1000, "Operating system:*" with a dropdown menu showing "Windows Server", "Erlang-C:" with the value 0.57, and "C Factor:" with the value 19.902. At the bottom of this green box is a "Calculate Baseline" button.

The application takes the first 4 fields from this server table and you must enter the data in the remaining 3 fields before calculating the baseline.

The operating system must be specified because the way Windows server report the Queue length is different from Linux and Z-Os. By the way, I included Z-OS because there are still an estimated quantity of 10,000 mainframes active in the world and mainly in very large enterprises like banks and insurance companies. Their sensitive applications still run on mainframe and many of them are written in COBOL. The last statistics show that USD 7.7 trillion in annual credit card payments, 29 billion annual ATM transactions and 12.6 billion transactions per day. And that's why 92 of the world's top 100 banks rely on mainframes to host their core systems.

For those who are too young to have known it, here is a modern mainframe:



So, back to the operating system, the Queue length reported by Windows server is just the single waiting line, while both Linux and Z-OS include the running transactions into their Queue figure. So, if you don't select the right operating system, the calculations will be wrong.

The application first estimates the probability of all cores being busy when a transaction becomes ready for processing, and this is done by using the Erlang-C formula which is the most accurate for this task. Here it is (just for fun):

$$P_w = \frac{\frac{A^N}{N!} \frac{N}{N-A}}{\left(\sum_{i=0}^{N-1} \frac{A^i}{i!} \right) + \frac{A^N}{N!} \frac{N}{N-A}}$$

I know it looks ugly, but if you break it down into its components, it becomes much easier to understand. Anyway, the model takes care of this for you. The next step is evaluating the Q component by inverting the M/G/n model that we will use to estimate Q length.

The M/G/n is

$$q = \frac{(1 + C) * B}{2(1 - B)}$$

It calculates Q time based on Erlang-C processor busy probability B and uses a factor named C to account for the distribution of services times and arrival rates. So, this is the formula we will use when we calculate a new Q time in the what if. But for the baseline, we actually need to invert the formula to extract the C factor from the measured data.

Again, this is all done by the application and you see the two values in the lower fields of the baseline.

Now, you can create any number of what if situations for this server. You can vary the Specint rating which is a relative performance index standardized for all processors, just like the MIPS were in the past. It is widely available but it's just a relative number, so we could very well decide to define a given processor as 1000 and attribute a value relative to that basis to all other processors. The Specint list is not for free, but I have a list of several thousands in my database.

If you change the Specint rating of the server in the what if frame, it will modify the CPU busy automatically. The you can also modify the number of cores. By the way, you can also just modify the % CPU busy without touching the Specint and the number of cores, to just model an increase or decrease of a given server, for example after modeling software changes with SCI.

Field	Baseline Value	What If Value
% CPU busy	85	56.7
Number of cores	8	8
Queue length	14	1.3
Specint	1000	1500
Operating system	Windows Server	
Erlang-C	0.57	0.11
C Factor	19.902	

In this example, we have just modified the Specint of the server while keeping the same number of cores.

Very easy to use, and always keep in mind that a model is an abstraction of reality (today we say a digital twin) and that the calculation applies to the measure situation, which is an average. Different situations will give different results, but the trend will remain the same. My point is that the 1.3 Q length is not a guarantee, but it will be in this range, which is 10 times less than the measure value.