Multi-threaded calculation in Excel, or “how calculation can become much faster in Excel 12”

One of the things I mentioned in my overview of Excel 12 post in September was that we had done some work to speed up calculation on modern hardware (multi-processor or dual-core chips). I thought I would take a brief break from tables to summarize that work and to see if any readers are interested in beta testing this feature.

Multi-Threaded Calculation To a large number of customers, Excel’s calculation speed is extremely important – perhaps the most important “feature” we ship. When planning Excel 12, we started a small investigation to look at different ways we could make Excel calculate faster on computers that had multi-processor or dual-core chips. The investigation turned out to be promising, so we continued the work, and the result is a very exciting feature that we refer to as multi-threaded calculation, or MTC. (Note – this is another working name, not a final name. Also note that our developers refer to this as MTR, or multi-threaded recalculation, but since most customers use the word calculation, I have decided to go that route for this post.)In a nutshell, this feature enables Excel to spot formulas that can be calculated concurrently, and then run those formulas on multiple processors simultaneously. The net effect is that a given spreadsheet finishes calculating in less time, improving Excel’s overall calculation performance. Excel 12 can take advantage of as many processors (or cores, which to Excel appear as processors) as there are on a machine - when Excel loads a workbook, it asks the operating system how many processors are available, and it creates a thread for each processor. In general, the more processors, the better the performance improvement.

Unsurprisingly, the design of the spreadsheet will have a direct impact on the size of any performance increase. At one end of the spectrum, a spreadsheet that has a lot of completely independent calculations (like a Monte Carlo simulation) should see enormous benefit. At the other end of the spectrum, a spreadsheet that has one completely linear set of calculations (where each formula depends on the previous formula, in linear fashion) will not see significant performance increases. The majority of the spreadsheets we see are in between these two ends of the spectrum, so we think that most everyone will see some amount of benefit from this work. Additionally, people who care about performance can tweak their spreadsheets to take advantage of this capability.We think this is really exciting work – we are still profiling and testing performance and making tweaks, but based on what we have seen so far, things like Monte Carlo simulations (which are almost ideal test cases) see almost perfectly linear speedup - that is, using 2 threads on a dual-processor or dual-core machine cuts calc time nearly in half – and the majority of other workbooks also see noticeable reduction in calculation time. This is especially exciting because of the increasing availability of dual-core chips in standard business desktop machines … simply installing Excel 12 on these machines should lead to calculation performance improvements for users.I can hear developers asking “what about user-defined functions (UDFs)?” The answer is that in Excel 12, we have enabled functions defined in XLLs to participate in multi-threaded calculation; VBA and automation add-in UDFs will not be multi-threaded (meaning, for example, that a VBA UDF and everything that depends on it cannot be processed simultaneously with other formulas). We have updated the XLL interface to allow developers to advertise their XLL functions as thread-safe (and to reference the big grid and a few other things to be covered in a later post about XLLs).A few other things about this feature:

- Calculation results are not affected by MTC, just the speed with which they are executed
- Users can turn the feature off if they so desire - with MTC turned off, Excel 12 will calculate the same way as it has in previous versions
- The first time a workbook is calculated on a machine with a different number of processors, there is some overhead incurred while Excel examines dependencies, so the maximum performance increase will be seen on the second and subsequent calculations (though there will still be improvement on the first calculation)
- Users can also manually specify the number of threads they want to run simultaneously, so if they want to run more than the number of processors on the machine, they can

Regarding that last point - you may be asking yourself why you would want to specify more threads than the number of processors - the reason we enabled running multiple threads per processor was to help out our customers who run workbooks that make a lot of external calls using XLL user-defined functions (UDFs). In certain industries, we see lots of customers that have slow-calculating spreadsheets where the bulk of the calculation time is spent outside of Excel in UDFs … the UDFs often run on other machines. For example, a customer may have a spreadsheet that makes 100 calls to a number of XLL UDFs on a server. The UDFs perform data retrieval or intensive calculations. Once all 100 UDFs have returned results to Excel, the spreadsheet then does some analysis on the results of those calls. Given that Excel 2003 is single-threaded, if each call to an XLL UDF takes 30 seconds, the model takes almost an hour to run, even on a dual-processor machine. If they ran the same spreadsheet on Excel 12, by default, the calculation time would be cut roughly in half, since Excel 12 would make 2 XLL UDF calls at a time – one for each processor. If the user then manually set the number of threads to 8, Excel 12 could make 8 XLL UDF calls simultaneously, meaning the calc time would fall from close to an hour to a matter of minutes, assuming the server can return the data at the same rate in either case.**Would you like to help?** The Excel team has reserved a number of positions in our upcoming technical beta for individuals that have the interest, time, hardware, and spreadsheets to help test the work we have done in this area (and all the other work we have done, although we are looking for a few people that are particularly interested in calculation performance). If you are interested in helping test this feature, please use the "Email" link above to contact me and answer the questions below. Before you do so, however, please take a look at a few caveats:

- Our technical beta is private, meaning you will be required to sign a non-disclosure agreement with Microsoft, and you will not be able to discuss the software with anyone outside the beta until the non-disclosure agreement is lifted at a later time
- You will need a dual-processor or dual-core machine running a recent MSFT OS to test this feature, and you will need to be comfortable installing beta software on the machine
- You will need to be able to devote time during the beta to running your spreadsheets and communicating issues/results to our development and support folks
- You will need a set of calculation-intensive spreadsheets to start with – the more variety, the better
- Since I have no idea what the response will be to this sort of request for help, we may end up with much more interest than we have room for, in which case I will pick folks randomly to participate … so sending me an email unfortunately does not guarantee we can give you a slot
- These positions are for individual testers, not companies. No problem if you work for a company, but we are looking to sign up individuals at this point, not entire departments.

If you have reviewed all that and are interested, please send me an email with the following:

- Some background on your company, your job/function, and how that involves Excel
- An idea of the hardware you have available for testing
- A description of the calculation-intensive spreadsheets that you have for testing – what do they do, what sort of features do they use, how long do they take to calculate, etc.
- An idea of how much time you could devote to testing per week over the course of a beta
- Whether you have already been nominated for Office 12 beta testing

Again, I can’t promise a spot to everyone that replies, so please don’t be disappointed if we don’t have space. And thanks in advance to anyone that does contact me.Next time, more on tables, I promise.