

Performance Implications of POWER7 Model 780's TurboCore Mode

Mark Funk
Rick Peterson

Abstract

The POWER7 Model 780 system offers an optional mode called TurboCore which allows the processor cores to execute at a higher frequency - about 7.25% higher - and to have more processor cache per core. Higher frequency and more cache often provide better performance. Describing what TurboCore really is and what it means to overall customer performance is the purpose of this paper.

DISCLAIMER NOTICE

Performance results are based on measurements and projections using standard IBM workloads in a controlled environment. This information is presented along with general recommendations to assist the reader to have a better understanding of IBM(*) products. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance or power savings improvements equivalent to the ratios stated here.

All performance and power data contained in this publication was obtained in the specific operating environment and under the conditions described within the document and is presented as an illustration. Performance or power characteristics obtained in other operating environments may vary and customers should conduct their own testing.

Information is provided "AS IS" without warranty of any kind.

The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Contact your local IBM office or IBM authorized reseller for the full text of the specific Statement of Direction.

Some information addresses anticipated future capabilities. Such information is not intended as a definitive statement of a commitment to specific levels of performance, function or delivery schedules with respect to any future products. Such commitments are only made in IBM product announcements. The information is presented here to communicate IBM's current investment and development activities as a good faith effort to help with our customers' future planning.

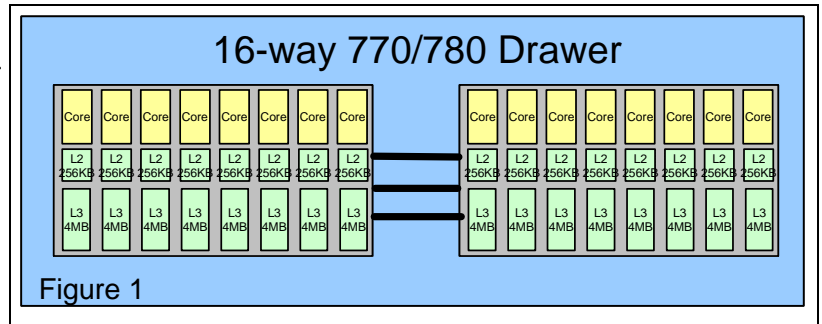
IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Commercial Relations, IBM Corporation, Purchase, NY 10577.

Information concerning non-IBM products was obtained from a supplier of these products, published announcement material, or other publicly available sources and does not constitute an endorsement of such products by IBM. Sources for non-IBM list prices and performance numbers are taken from publicly available information, including vendor announcements and vendor worldwide homepages. IBM has not tested these products and cannot confirm the accuracy of performance, capability, or any other claims related to non-IBM products. Questions on the capability of non-IBM products should be addressed to the supplier of those products.

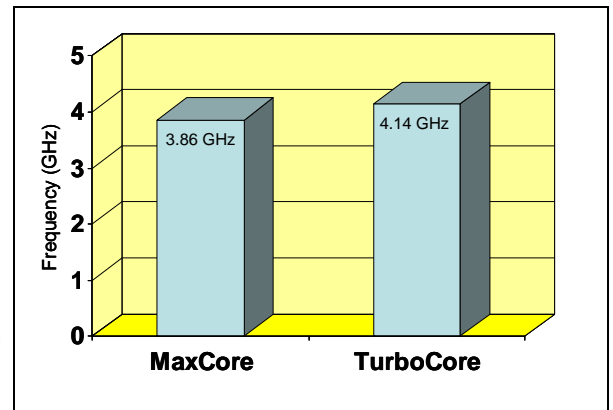
Overview

Every POWER7 Model 780 system uses a building block consisting of two-chip drawers. Each of these processor chips is capable of enabling eight active processor cores. As seen in Figure 1, each drawer is capable of supporting up to 16 processor cores. Each core is packaged with its own cache.

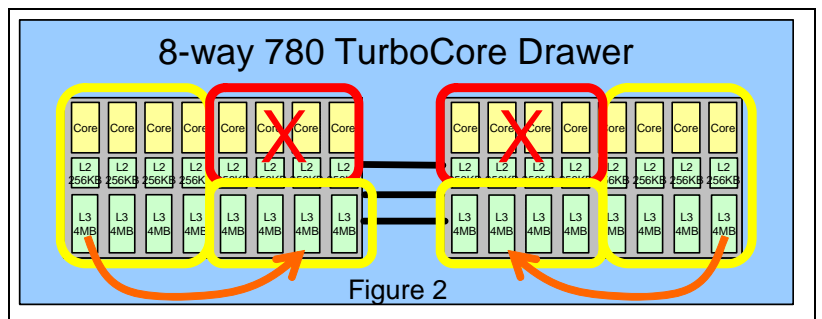
TurboCore is a special processing mode of these systems wherein only four cores per chip are activated. With only four active cores, ease of cooling allows the active cores to provide a frequency faster (~7.25%) than the nominal rate.



The POWER7 cache design has 4 Mbytes of L3 cache per core. Although it may look as if there is a private L3 cache per core, this cache can be shared between cores. The cache state from an active core's L3 cache can be saved into the L3 cache of less active cores. In the case of TurboCore, as can be seen in Figure 2, the cache state from the four active core's can be saved into the L3 cache of TurboCore's inactive cores. The result is more accessible cache per core.



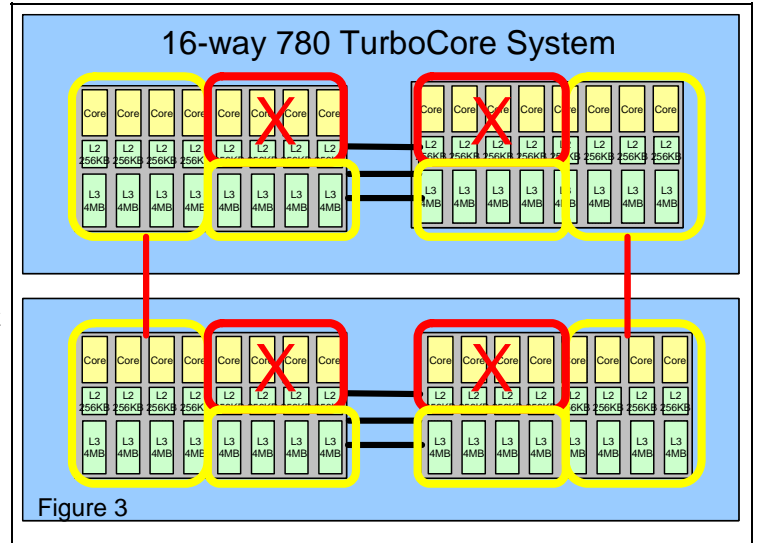
Both the higher frequency and the greater amount of cache per core are techniques for providing better performance. It is not uncommon for a longer running, even multi-threaded, workload accessing largely private data to see a performance benefit well in excess of what might be expected from the better frequency alone. Even more complex workloads residing in a partition scoped to the cores and memory of a given processor chip can see similar benefits.



But it is important to note that in doing so, the processor chip's core count has decreased from eight cores per chip to four. An 8-core partition formally residing on one processor chip now must reside on two. A system needing sixteen cores and packaged in a single drawer as in the earlier figure requires two drawers when using TurboCore as in Figure 3.

An interesting effect from having doubled the number of chips is that the extra chips and extra drawers allow the maximum amount of memory per core to also double. Whether TurboCore or MaxCore, each chip's memory controllers directly drive up to 8 memory DIMMs.

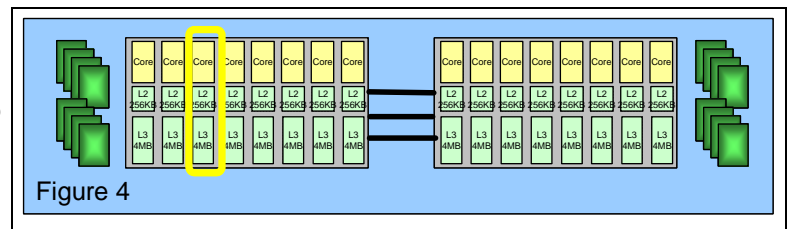
Another effect, though, stems from the fact that chip crossings introduce extra time to storage accesses. For the same number of cores there are often more chips required with TurboCore. So more chips often then implies a higher probability of such longer latency storage accesses. We'll be looking at this performance effect next.



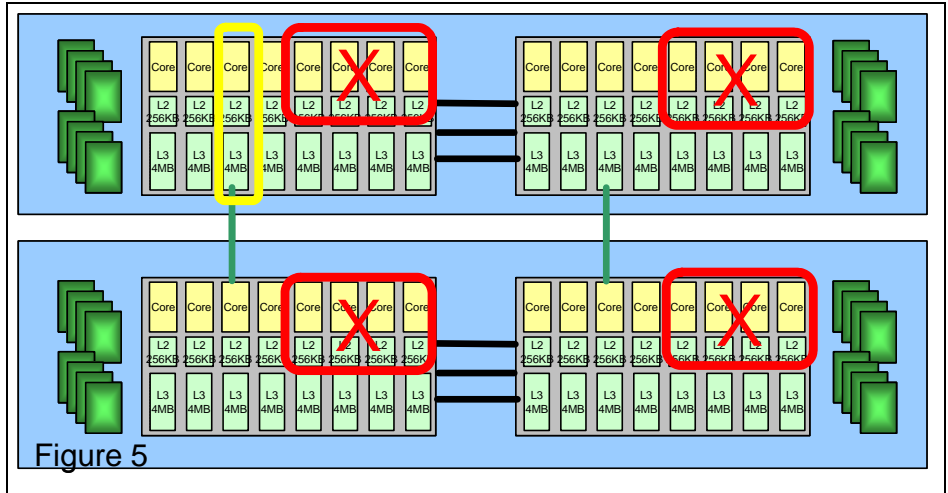
Background

Memory Access Latencies:

We start here by introducing a new concept. A processor core incurring a cache miss often needs to access main memory to get the data that it wants. It takes quite a considerable number of processor cycles to access memory -- that is why cache exists in the first place, to provide faster access to data which was recently accessed for some other purpose. As in the single-drawer, two-chip system shown in Figure 4, memory is attached to each chip. From the point of view of some reference core, there is memory attached to its own chip and memory attached to some other chip; all of it is accessible from any core. It nominally takes about 1/3 longer to access the memory attached to another chip than it does to access the locally attached memory. Both local and remote memory accesses are "fast", but to get the most capacity out of a system, you would prefer that the large majority of the memory accesses were from local memory. It happens that the IBM i operating system knows this and is often successful in arranging for the data to be local to the core doing the memory access. How well the operating system does this management, though, is dependent upon customer workload as well as how the partition's cores and memory are assigned to these chips.



We next look at another 16-core system, this time - as in Figure 5 - with two drawers using TurboCore mode. From the point of view of the same reference core, there still remains some amount of local memory, but the remainder of the memory is attached to three other chips, two of them in another drawer. With randomly distributed pages, previously over two chips and in this example over four, the probability of an access from the reference core's local memory drops from 50% to 25%. Of the remote memory accesses, for the two chips directly connected to the reference chip, the extra latency to remote memory is still about the same. Notice, though, that the third chip requires an extra hop to access its memory and so a still higher access latency. But it happens that there are many reasons that pages are not randomly distributed. The actual distribution (which tends toward local accesses) is dependent upon customer workload type and upon how partition's core and memory resources are allocated.



There is another factor that adds latency for cross-chip accesses. As the utilization of the system increases, often times so does the usage of the links between the chips. Whenever a link happens to be busy, any subsequent need to use the link must wait slightly longer for its use of the link. It is possible, though, that the cores could be very busy - say accessing their own memory - and the links are not. In such cases, there is little extra latency resulting from busy links. But if the high system utilization also results in high utilization of the links, inter-chip storage accesses can take considerably longer than a nominal latency.

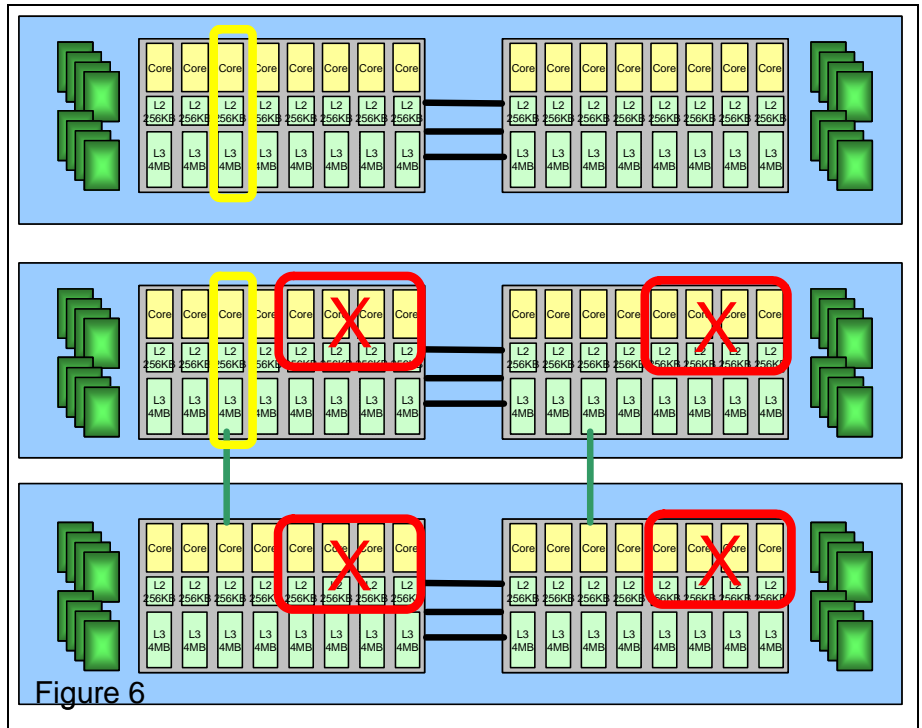
Cache Access Latencies:

In SMP systems like these, just as with the contents of all main storage being accessible from all cores, the contents of all cache is also accessible from all cores. This is true whether the cache being accessed is associated with a core on the same chip as a reference core or on a different chip. And, just as with main storage accesses, a cache fill from a local core's cache is faster than an access from another chip's cache. Relatively speaking a remote cache access is considerably longer than that (recall that main storage accesses are nominally 1/3 longer).

So now again consider two 16-core systems (Figure 6), one using two chips in one drawer and all eight cores per chip and the other using TurboCore mode and so requiring four chips over two drawers and its four cores - and extra cache - per chip. For a workload in which a reference core is not frequently accessing the contents of another core, this difference in cache access latency is not particularly pertinent.

For a workload in which a reference core is frequently accessing the cache of another core, if the cores that are being accessed happen to reside on the same chip as the reference core, again the difference in cache access latencies is not much of an issue.

But here the difference in location of the cores resulting from the use of TurboCore can matter. With eight cores per chip, nearly half of the system's core's cache is on the same chip as a reference core. With TurboCore's four core's per chip, nearly 3/4 of the cache resides on a remote chip.



As with memory accesses, the IBM i operating system knows of this effect and can offer ways to create greater locality, both of memory and of the chip's cache. Again, though, how well the probability for local cache access can be influenced is a function of the customer workload and of the placement of each partition's cores and memory.

Summary Before Continuing

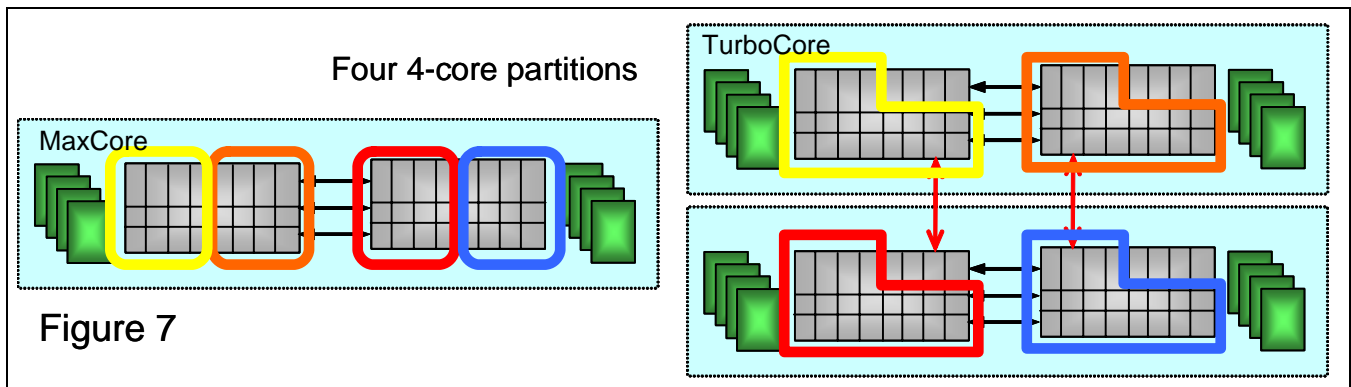
As you have seen, TurboCore provides a higher frequency core and more cache per core. These are normally very good things for performance. You also now know that to make use of these positive attributes, the system's active core placement needed to change from having eight cores per chip to having four cores per chip. So using TurboCore can also mean a change in the number of chips and perhaps in the number of drawers. For a given partition, this in turn can mean an increase in the probability of longer access latencies to memory and cache. But partition placement and workload type can influence these probabilities. As a result, the positive benefits of TurboCore's higher frequency and increased cache also have the potential of being offset to various extents by these longer latency storage accesses. In cases in which the cross-chip accesses are relatively limited, all of TurboCore's benefits can remain. We'll be looking at this more in subsequent sections.

Performance Implications for Applications with High Data Sharing

So when would you expect that using TurboCore mode would produce the best results when compared to MaxCore? The comparisons to come will all be comparing these modes using the same number of cores to ensure the same software licensing costs.

Best Results:

Consider four partitions, each with four cores per partition. Let's also assume that the main storage size requirements for all four is equal so that the memory used by the partitions can be allocated local to the chip holding each partition's cores. The results are as shown in Figure 7.



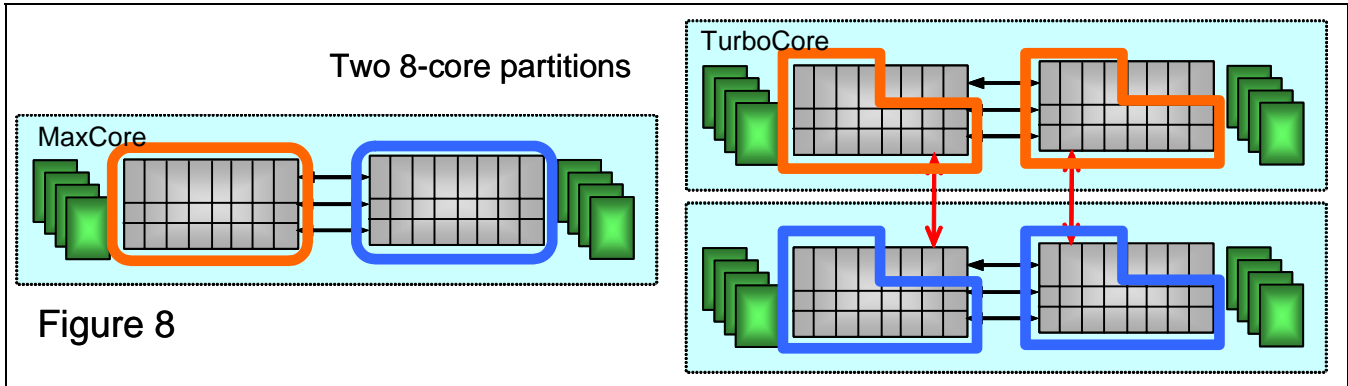
Here, no matter the type of data sharing which is going on inside of each partition, the cache and main storage accesses are made local to each chip. All cache and memory accesses are made with the latencies of strictly local accesses. The TurboCore-based partitions are all seeing the benefit of the higher frequency and the larger cache. But there are two other often significant benefits.

1. Because each chip has its own memory controllers and busses to local memory, each partition sees this benefit as well; this higher bandwidth also means consistently good memory access latencies even as the workload on these partitions grows to its maximum.
2. Each chip supports some number of DIMMs slots. Four processor chips also means the opportunity to have twice as much memory as compared to MaxCore.

The performance of TurboCore using this basic notion of keeping partitions local to sockets is expandable to any number of chips and drawers.

Good (potentially Very Good) Results:

Here we increase the size of the partitions to eight cores. Using the same number of drawers as shown in Figure 7 and again assuming equal memory being used, two 8-core partitions are packaged as shown in Figure 8.

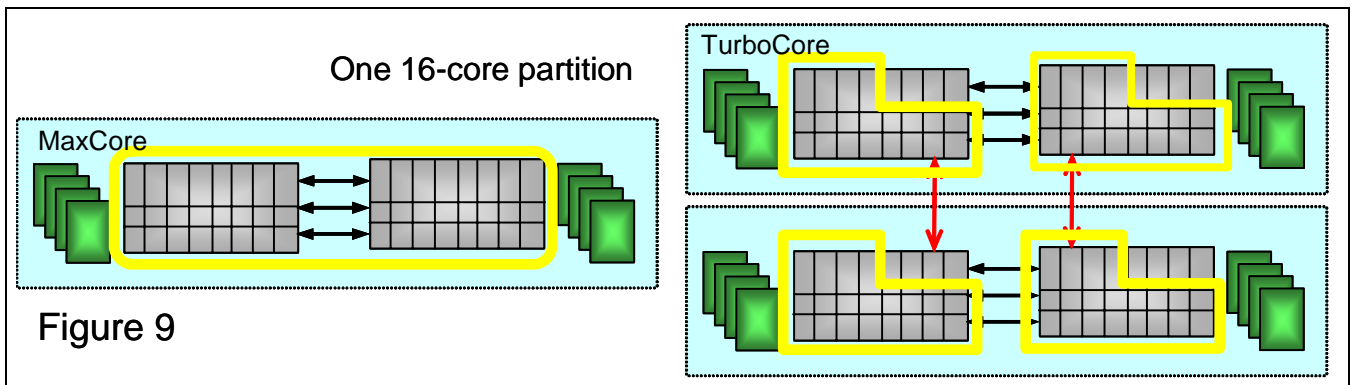


In this case, the MaxCore partitions each have all the benefits of strictly local memory access, but not the higher frequency and larger cache per core of TurboCore. TurboCore mode here also sees some benefit from the greater bandwidth to memory. But there is often some considerable inter-chip cache and memory access traffic adding some extra time to the storage accesses.

It is for this reason that the partition's of Figure 8 often will not see the full benefit of TurboCore as experienced by the partitions of Figure 7. But under some circumstances they can approach this better performance. If the OS of each partition can discover that the data can be placed close to where the work is executing on the cores, the probability of cross-chip traffic can be minimized, allowing a quite considerable opportunity to experience TurboCore's benefits.

Equivalent Results:

Here we assume a single large partition. For this example, Figure 9, we are using a 16-core partition.



In both of these, the partition is spanning multiple chip boundaries. The partition on both cases is experiencing some level of the added latency associated with cross-chip storage accesses. As in all of these, the partition using TurboCore is experiencing the benefits of the higher frequency and larger cache. But the TurboCore partition is also often experiencing a higher probability of cross-chip latencies. As discussed earlier, when these higher access latencies to storage occur, they have the effect of offsetting the benefits of TurboCore.