The P6 for Servers

This section discusses the features designed into the P6 microarchitecture and into the P6 implementation to ensure that a P6 -- actually multiple P6's -- can be used to design a cost-effective, high-performance server product. The combined impact of these design features will drive the P6's wide adoption as an enterprise solution.

Agenda

- **What problem are we solving**
- **P6 implementation yields additional Server benefits**
  - Two die in package with private cache bus & external system bus
  - P6 chipset partitioning designed for scalable servers
  - Reliability features integrated into P6 and chipset

Let's first take a step back and look at the key system problem we set out to solve with the P6 processor - its interaction with system memory. Once the problem is understood and our method of solving it is known, we will look at the P6 implementation issues specifically designed to benefit servers.

What problem are we solving

- P6 will make a **GREAT** Server Building Block
- Back to Basics
  - Root of problem is characteristics of server programs
  - Server CODE is BIG, Server DATA sets are BIG
  - Results in a potentially large L2 cache miss rate
  - CPU could spend a lot of time WAITING

Long Memory Latencies Slow System Performance

In designing the P6, we analyzed a lot of Pentium® processor-based systems and discovered that the CPU was spending a lot of time WAITING for memory. Once it had the instructions or data, it was very fast, but it was difficult to feed it fast enough.

Once an address (A) strobe was issued, the Pentium processor waited for it's data (D) to be available before proceeding with execution. Thus, the processor STOPS while waiting for memory.

How can we solve this long memory latency dilemma?

Solving the long latency problem

- Traditional “Brute Force” Approach
  - Huge caches, more memory
  - Increases server entry price
- P6’s Innovative Approach
  - Execute other instructions OUT-OF-ORDER while waiting
  - Can get same, or greater, performance at lower system cost

In this code fragment assume that (X) is a cache miss.
Therefore, CPU will stall waiting for memory to respond.
BUT, CPU could carefully execute c: and/or d:
(NOTE: Cannot execute b: since it is dependent on a:).

```
a: r1 = Mem(X)
b: r3 = r1 + r2
c: r5 = 9
d: r4 = r4 + 1
```

P6 Approach will set a new Price/Performance Point

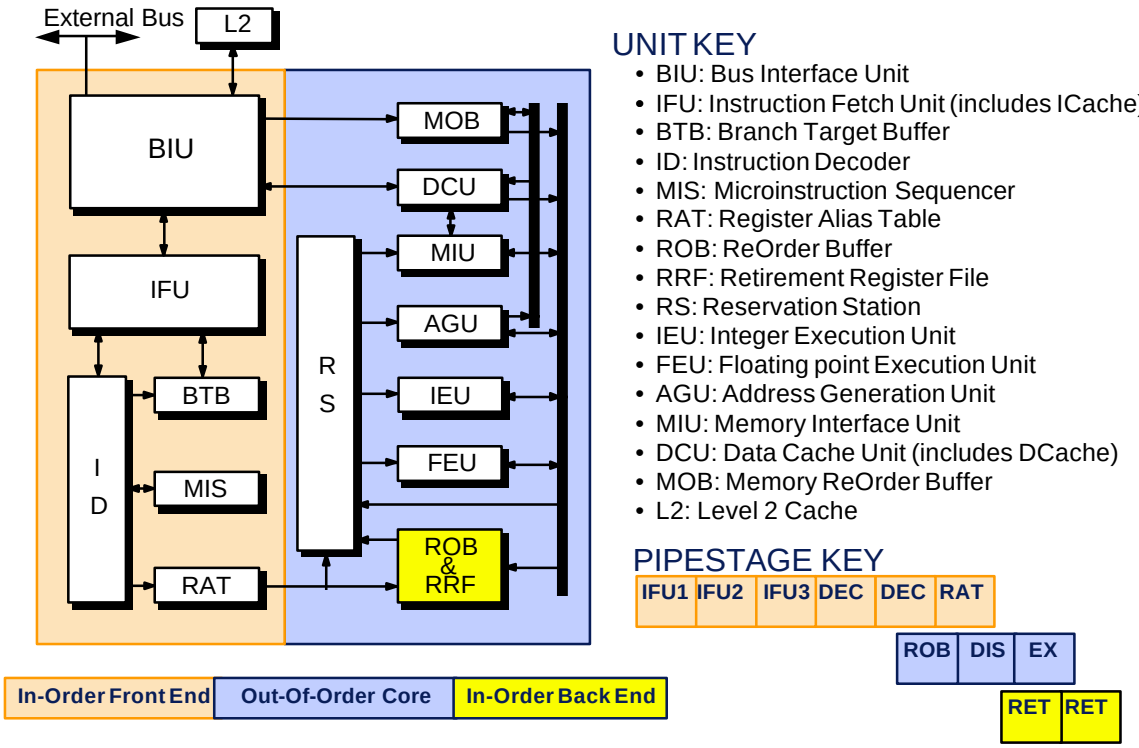
Currently, high-end servers implement the obvious approach to reducing main memory latency -- using a huge L2 cache (i.e., 1MB to 4MB). Although this is very effective, it also very expensive.

The P6 took an innovative approach. While waiting for memory to respond, the P6 does other useful work. That is, the P6 speculatively executed instructions AFTER the cache miss, since program flow is likely to need these anyway. BUT it is careful NOT to commit these to program state (registers and memory) until the Instruction Pointer reaches these instructions.

In this example, we speculatively execute c: and d: while waiting for instruction a:

Techniques such as this enable the P6 to execute more instructions with the same speed (price) memory system.

P6 Internal Block Diagram



IFU does all the prefetching, and predecode, in pipestages IF1, IF2 and IF3

ID decodes the instructions into micro-operations (uops), in pipestages DEC,DEC

RAT allocates destination registers for each uop

ROB determines uop dependencies and schedules execution using RS

DIS, uops are dispatched to AGU, IEU, FEU, etc once dependencies are resolved

EX, uops are executed

RET, instructions are retired in original program order

For more information on the internal operation of the P6 Microarchitecture refer to section on ‘How the P6 Works’.

Benefits of this new architecture

- Do not WAIT on a cache miss, continue to do work
- Memory latency is hidden, other work started
- In steady state 3-4 micro-operations enter at the top and 3 micro-operations exit at the bottom

- What prevents a constant 3 micro-ops per clock?
  - Mis-predicted branches, must flush all work
  - Mis-aligned data (i.e. across two cache lines)
  - Serializing instructions (e.g. IN/OUT)
  - Interrupts/faults/context switches

The P6 microarchitecture was designed to process 3 uops per clock continuously. The three internal units are balanced to maintain this high throughput.

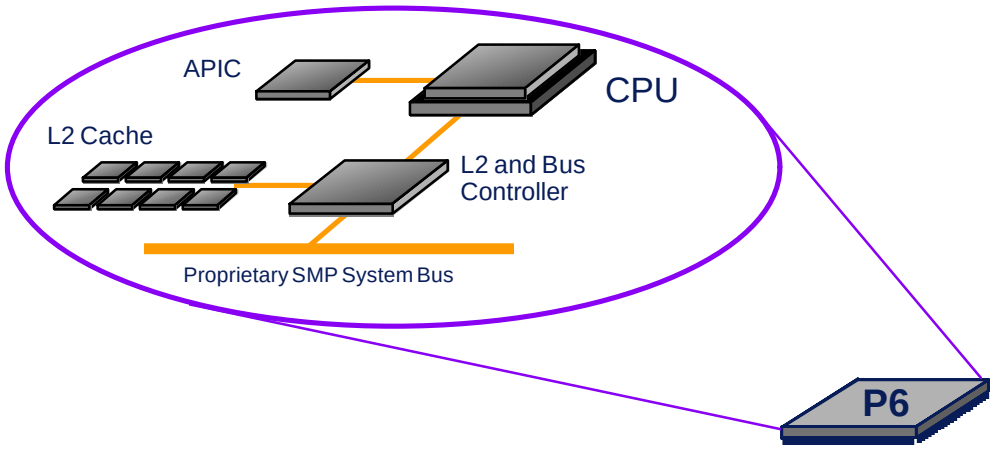
Agenda

- **What problem are we solving**
- **P6 implementation yields additional Server benefits**
  - Two die CPU with private cache bus and external system bus
  - Reliability features integrated into P6 and chipset
  - P6 chipset partitioning designed for scalable servers

Having discussed the key problem the P6 addresses, we now move into server implementation issues.

The P6 Component

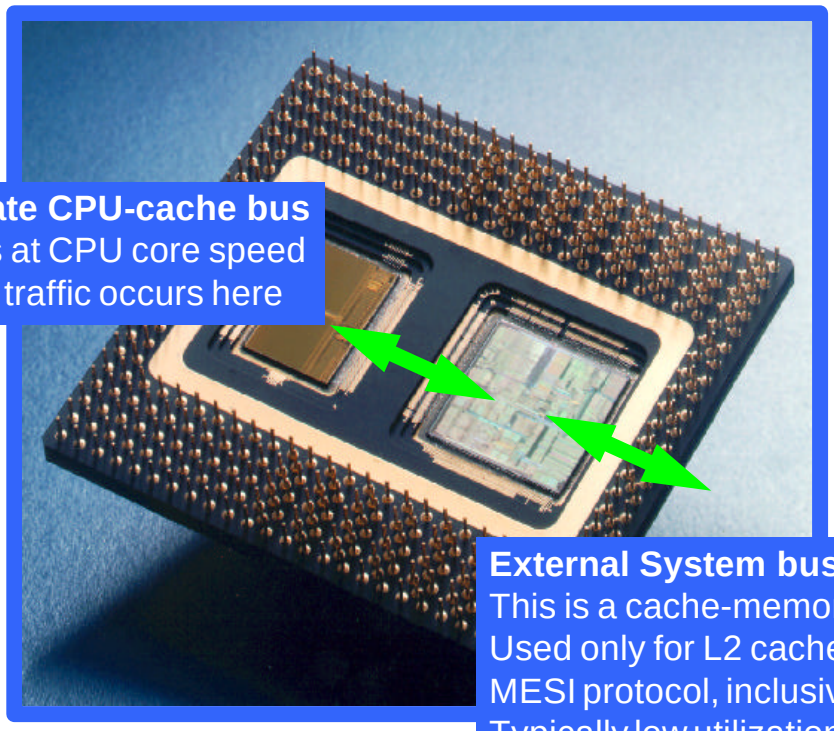
- **Integrate CPU-cache-bus complex in a single package**
  - **Ease of upgrade, performance, reliability**



Integrating the CPU, L2 cache, APIC and Bus controller into a single package makes the P6 a unique multiprocessing building block element. The P6 bus is a cache-coherent, transaction oriented bus that scales to four processors even under the worst benchmark loading scenarios (TPCs). Also, the board area taken by the P6 is less than a Pentium® processor plus it's L2 cache.

This integrated design means that multiprocessing server designs using the P6 will be commonplace. This also means that all desktop systems using this P6 will have an L2 cache and will thus perform very well.

P6 CPU has two independent buses



Most of a program's execution occurs within the P6 package. This means:

-All of the CPU-cache traffic runs at full core speed on a private 64bit bus, which includes error checking and correction (ECC).

-L2 cache misses leave the P6 package on the 64bit external bus (with ECC). This bus runs at 1/2, 1/3 or 1/4 of the core speed depending upon the system design.

-External bus includes MESI protocol and multi-master arbitration; no other components are needed to support 4 interconnected P6 processors. This capability has been called 'glue-less MP'. Multiple CPUs communicate using MPS protocol on an APIC interconnect bus.

-External bus utilization is low even on benchmarks that miss the cache a lot. This means that there is 'spare' bus bandwidth to add more processors or I/O.

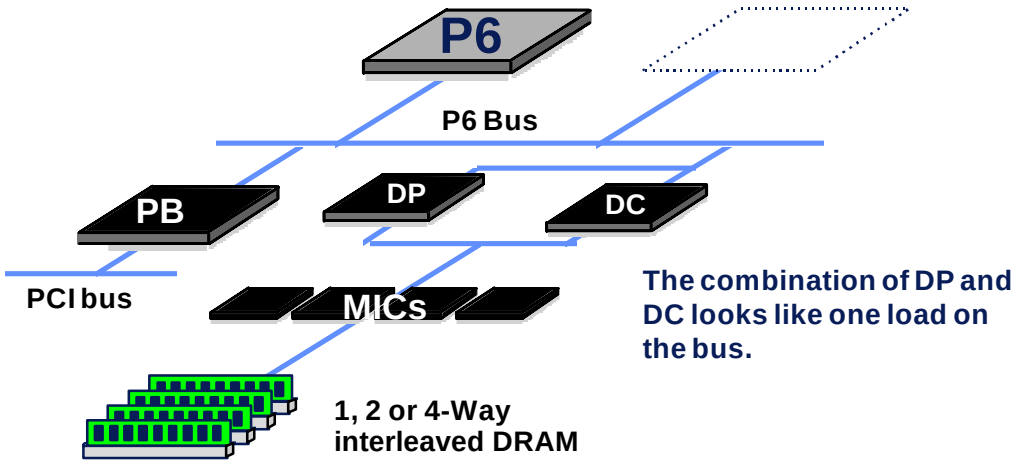
Why the P6 is Good at TPS

TPS characteristics	P6 attributes
Integer Performance starved	High SPECint92 figures
I/O Hungry	High bandwidth transaction-oriented bus allows code/data fetch to continue while waiting for I/O
Large data set	4 Mbyte data pages, Non blocking data access hide the latency of cache misses
Poor code locality	4 Mbyte code pages. Large on chip 4 way set associative L2 cache.
Code with frequently taken branches	Large BTB and Adaptive Branch Prediction, with the static branch predictor as a backup. Large on chip L2 cache.
Instruction level parallelism small	Dynamic Execution exposes more parallelism across multiple basic blocks

The P6 is particularly good at transaction processing due to it's new micro-architecture. And this hardware (i.e., CPU and bus) scales very well to 4 P6s.

(TPS = Transactions per second.)

P6 Chipset Interconnection



PB - PCI Bridge
MC-DP - Memory Controller Data Path
MC-DC - Memory Controller DRAM Control
MIC - Memory Interface Controller

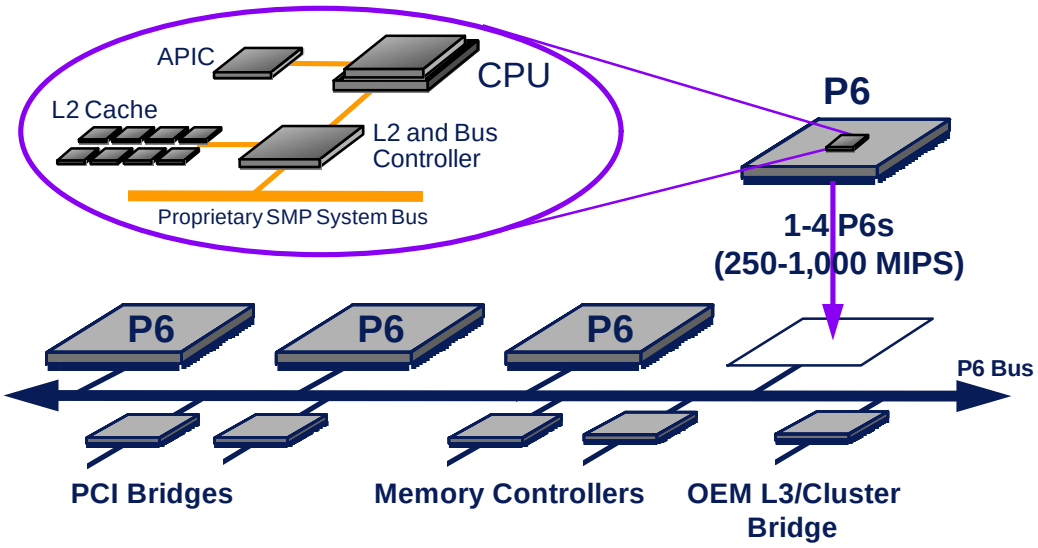
The P6's chipset was designed alongside the P6.

The chipset consists of a single chip interface to the PCI bus (PB = PCI Bridge) and a high performance memory subsystem controller (MC = Memory Controller).

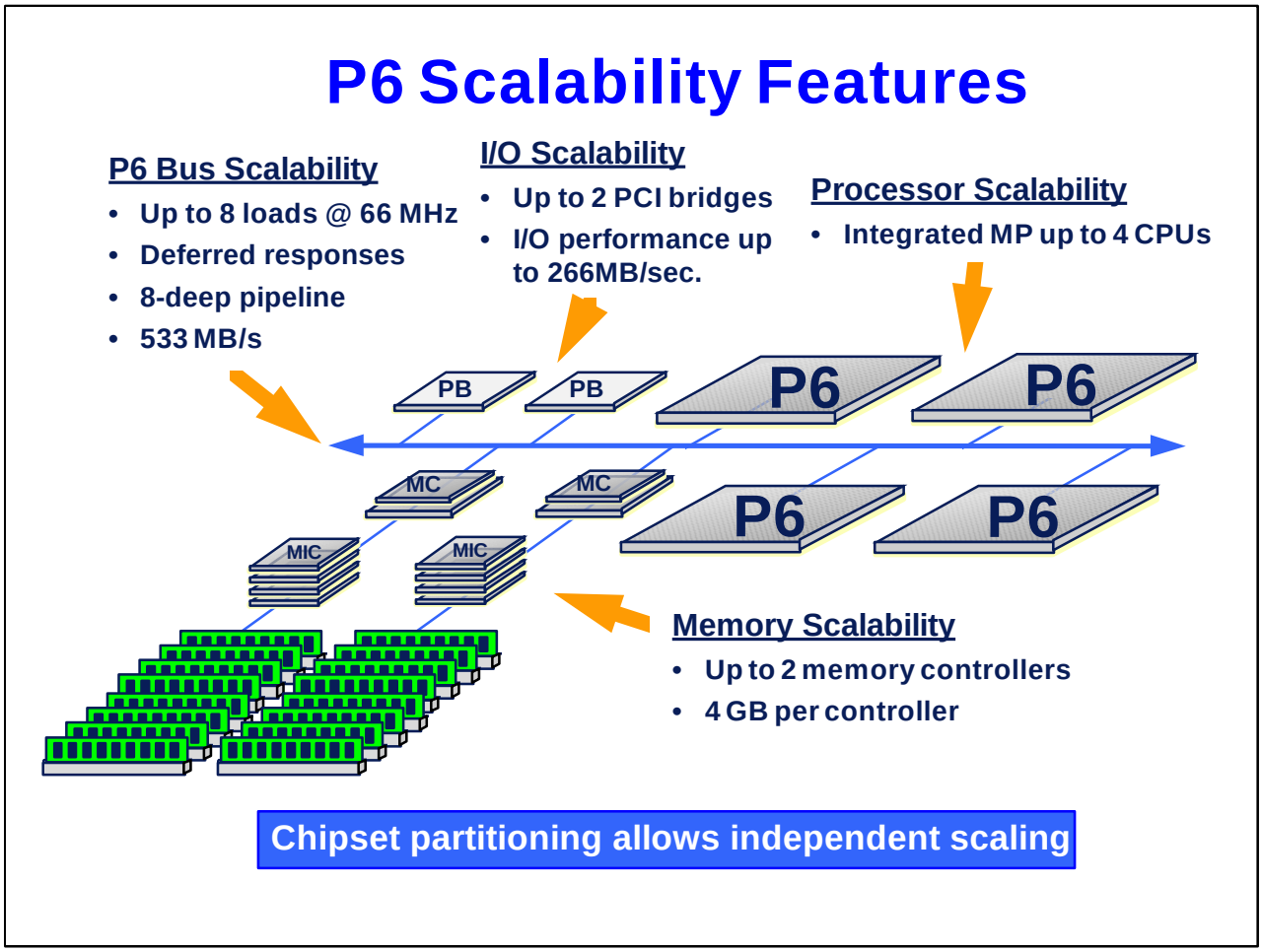
The MC is implemented as a Data Path, Data Controller and four Memory Interface Controllers which enable 4-way interleaving.

P6-based Servers are SCALEABLE

- Integrate CPU-cache-bus complex in a single package
  - Ease of upgrade, performance, reliability
- The CPU bus becomes a system bus
  - Imagine a packet switch connecting CPUs, I/O, & memory
- Chipset partitioning allows independent scaling of I/O and memory
  - Suitable for a wide range of system implementations



The P6 chipset is designed to allow independent scaling of CPU, memory and I/O.

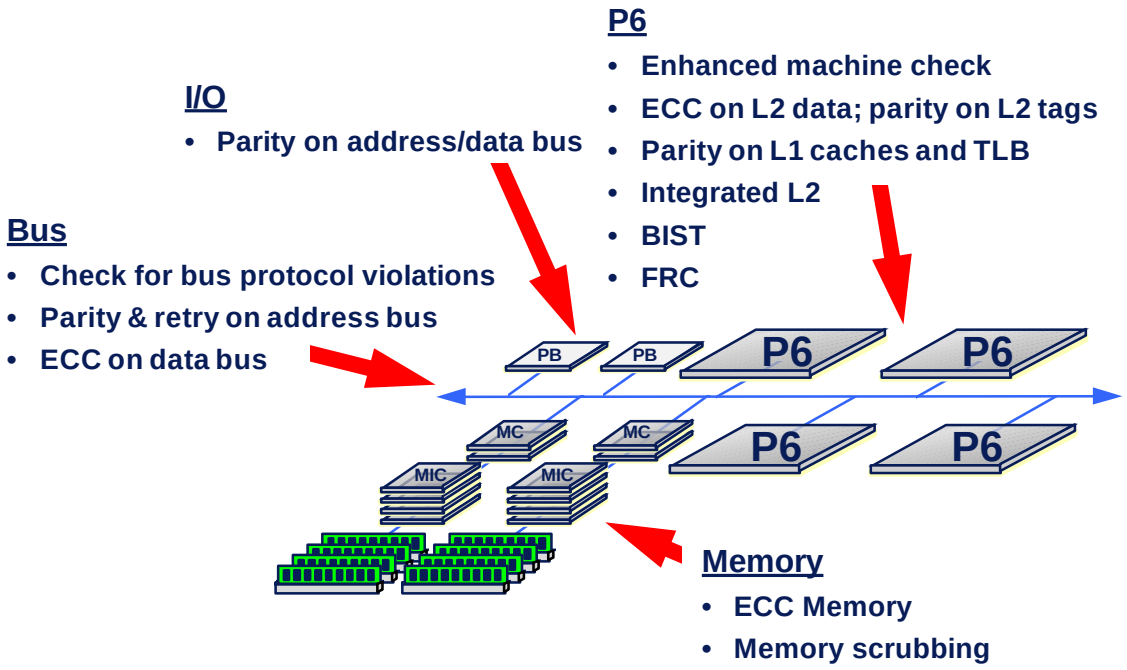


The platform design goal of SCALABILITY is well-served with the P6 and chipset implementation.

Key features are shown above.

(Note that MC and MIC components DO NOT stack -- They are shown this way for clarity.)

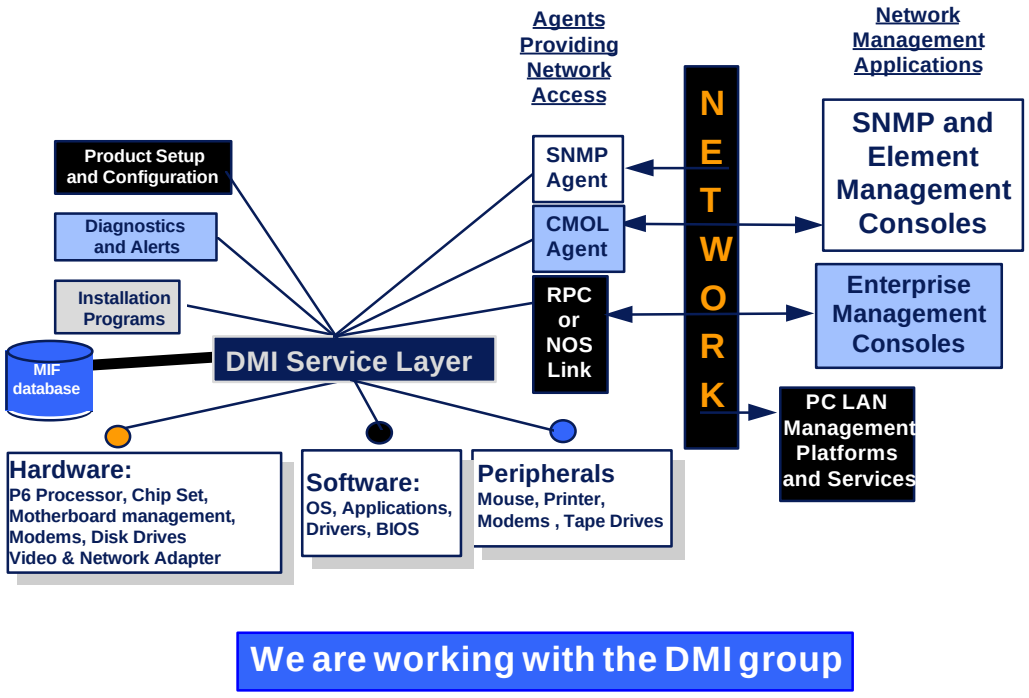
P6 Reliability Features



Features included as part of the fundamental product

A key design objective of the P6 and its chipset was to include the features required as a foundation for reliable system design.

Providing a standard System Management Interface



The software interface to these hardware features is DMI.

P6 Delivers for Servers

- **Performance**
- **Reliability**
- **Scalability**
- **Manageability**

In sum, the P6 design has always sought to address the unique needs of server applications. The P6 architecture delivers great server performance, includes new reliability features, facilitates scalability, and fits with efforts to improve systems management.