You've been hired by a new startup company to custom build them a new multicore processor for their killer application. You have a team of architects and engineers at your disposal who can design the processor if they only knew how many cores are needed. There's a limited budget and you need to keep costs down.

You determine that 60% of the tasks in the application can run in parallel, and that the work can be uniformly divided among them. If your design team can put a maximum of 1000 cores on a chip, how much of a speedup can you expect? What other factors might you consider in determining how to build your processor?

We assume that each core does not implement functionality that cannot be practically implemented. Each core is 2-4 issue processor, with a reasonable pipeline depth, cache or local store, and bandwidth.

From Amdahl's Law, we can determine that the maximum potential speedup is

$$speedup \leq \frac{1}{(1-p) + \frac{p}{n}}$$

where $p = 0.60$, the parallel fraction of the work; and $n = 1000$, the number of processors available for computation. We approximate the second term to be negligible, and hence the speedup is less than or equal to (1 / 1 – 0.60) or 2.50x.

Notice that for $n = 10$, the speedup is already 2.17x, and the additional 13% gains in performance from an order of magnitude increase in cores appears far from justified.

With fewer cores, more time can be spent on optimizing the performance of each core to better match the target application. For example each core can implement specialized datapaths for exploit bit-level parallelism, or custom instructions to provide an application-specific instruction set. This leads to a heterogeneous multicore architecture that potentially affords a greater speedup by leveraging the same concepts used for ASICs (application specific integrated circuits).