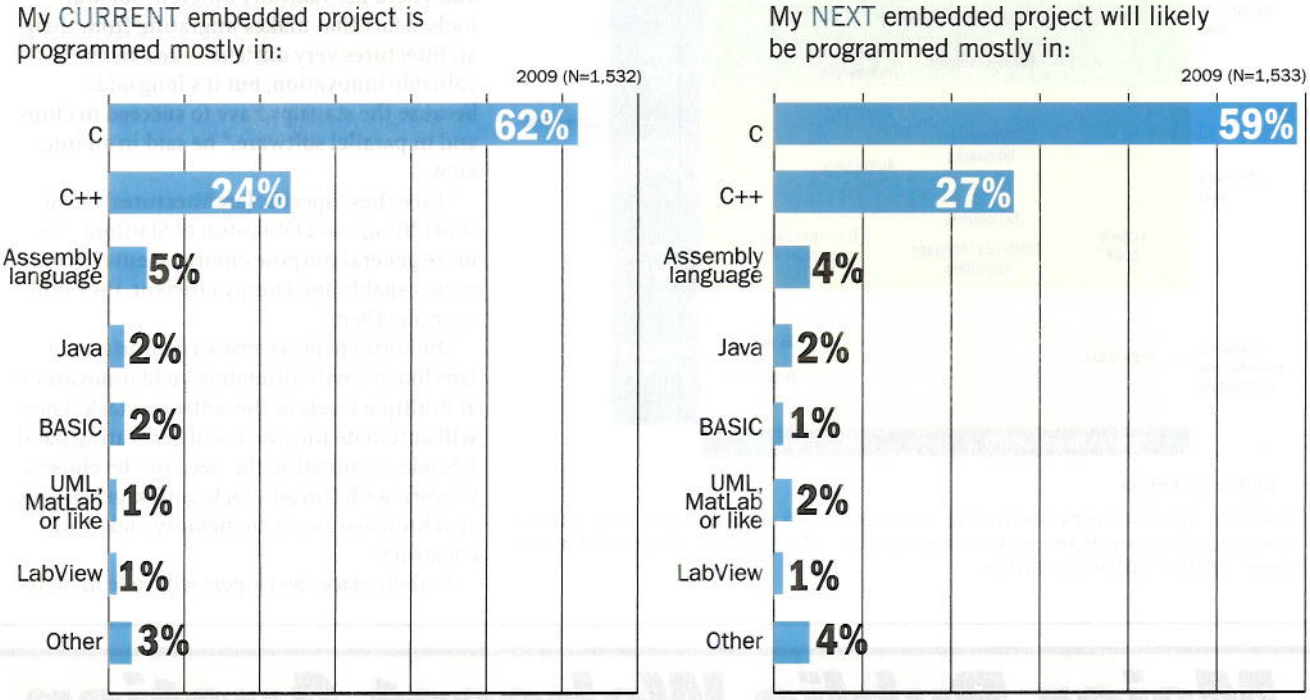
C language dominates today's and tomorrow's sequential programs



SOURCE: TechInsights' 2009 Embedded Market Study

Meanwhile, OEMs are already finding their own ways to address the problems.

Telecom architect Bachmutsky said control plane designs are adopting system-level symmetric multiprocessing tools to harness multicore chips using an SMP operating system. The resulting designs "look [to software] like systems with multiple line cards and a load balancer that divides traffic between blades," he said.

Data plane designs that require tenfold greater performance are tougher because they often use assembly language coding. That environment cannot afford the shared memory overhead of SMP constructs.

Developers find themselves carefully dividing up tasks to each core, watching for data dependencies. They must craft detailed messaging schemes between tasks, then figure out ways to communicate between the data and control plane software stacks, Bachmutsky said.

With the assembly language code, "you are closely linked to the silicon provider and their libraries, and you cannot port software easily to another processor," he said. "Whatever you choose, you wind up married to those solutions."

Progress at the bleeding edge

Some specialized apps are moving even further down the road to parallelism, albeit using proprietary chips and tools. For instance, Nvidia has been pioneering work in massively parallel programming using versions of its graphics chips with its Cuda environment in a range of vertical apps, such as oil and gas exploration.

Some designers report success living on the bleeding edge of parallel processing. For example, England's Cambridge Consultants has done contract design work on 3G and WiMax basestations using devices from PicoChip (Bath, England) that pack 250 cores per chip.

For those applications, the consulting company has actually found the PicoChip devices a better approach than some quad-core digital signal processors. "It seems strange to people at first, but we get shorter, more reliable programs with higher-quality output [using PicoChip] than with conventional single or low-core-count DSPs," said Monty Barlow, who leads a DSP group at Cambridge.

"The high [core count] multicore architecture lets you split functions between cores, develop and test them separately and then move on to other parts of the system knowing those parts will not interact in negative ways," said Barlow. "The alternative is to write programs as threads and rely on an operating system to share time, but the tasks run at unrelated rates, and one day things may conspire against you and something will run late," causing a crash.

The approach requires rewriting software for the PicoChip devices. But Barlow said he finds it a worthwhile trade-off doing more work up front on the architecture so that the