

## Re: future of programming languages

**Source:** <http://coding.derkeiler.com/Archive/General/comp.programming/2004-03/2574.html>

---

**From:** Omri Barel (*spammer\_at\_p2z.com*)

**Date:** 03/31/04

Date: Wed, 31 Mar 2004 17:20:31 +0100

Alan Balmer wrote:

> *On Tue, 30 Mar 2004 22:19:54 +0100, "Malcolm"*  
> *<malcolm@55bank.freemove.co.uk> wrote:*  
>  
>  
>> *"Ben Pfaff" <blp@cs.stanford.edu> wrote in message*  
>>  
>>> *I'm curious what you mean by "computers will have no CPUs."*  
>>> *Can you elaborate?*  
>>>  
>>  
>> *A typical computer has an instruction stream which passes through a limited*  
>> *number of registers or logic gates. Though nowadays there is often some*  
>> *parallelism, conceptually instructions are still executed in sequence.*  
>> *A nervous system doesn't work like that. Each nerve input is processed by*  
>> *neurons and output produced in a massively parallel system. Even when you*  
>> *remove portions of the brain, processing is often only slightly degraded.*  
>> *Nervous systems can do things which it is very difficult to program computers*  
>> *to do, but are of real use away from the protected environment of*  
>> *pre-formatted input and defined output that most programs operate in.*  
>> *Recognising individual people by a visual image, for example, even if the*  
>> *face is partially obscured by a coffee mug.*  
>>  
>  
> *First, there are indeed computers which process instructions in*  
> *parallel, even conceptually. They aren't even uncommon now.*  
>  
> *Second, you didn't answer the question, unless you're trying to say*  
> *that "multiple CPUs processing in parallel" are not CPUs.*  
>

I think he means a system where no processor is central. When you have 65,000 processors, you can \*call\* each of them 'central processing unit', but none of them is really central.

How do you program 65,000 processors, when they don't all do the same thing? Do you maintain 65,000 C programs? How do you distribute this

## comp.programming: Re: future of programming languages

software among 65,000 processors? What happens if 102 of them are dead when you distribute the software, and you replace them only later (while the others are alive)? What happens if you can treat processors separately as well as together (assign 4 processors a task, and then assign each a different task) – do you code all this in C?

The same questions apply to a massively distributed project as well. Today, massively distributed projects consist of a huge number of 'clients' all running the same software with different data, reporting to one (or several) central servers. What happens if your software is so complicated that each client should run different code (which may be changed at run-time as a result of other clients' output)? Do you code it in C?

Most languages do not directly support the notion of multiple processors. You can start a few threads and hope that the hardware will do the right thing, but controlling 65,000 (different) threads in one C program sounds like a nightmare. The (future?) hardware may provide direct support for synchronisation between processors. If you program each processor individually you might not even be able to take advantage of such hardware support.