C++ and Beyond 2011, Banff

Welcome!

C++ and Beyond 2011, Banff

Welcome!

Why C++ ?

power: driver at all scales – on-die, mobile, desktop, datacenter

size: limits on processor resources – desktop, mobile

experiences: bigger experiences on smaller hardware; pushing envelope means every cycle matters

1979-1989

Research: C with Classes, ARM C++

1989-1999

Mainstream

C++ goes to town (& ISO, & space, &c)

1999-2009

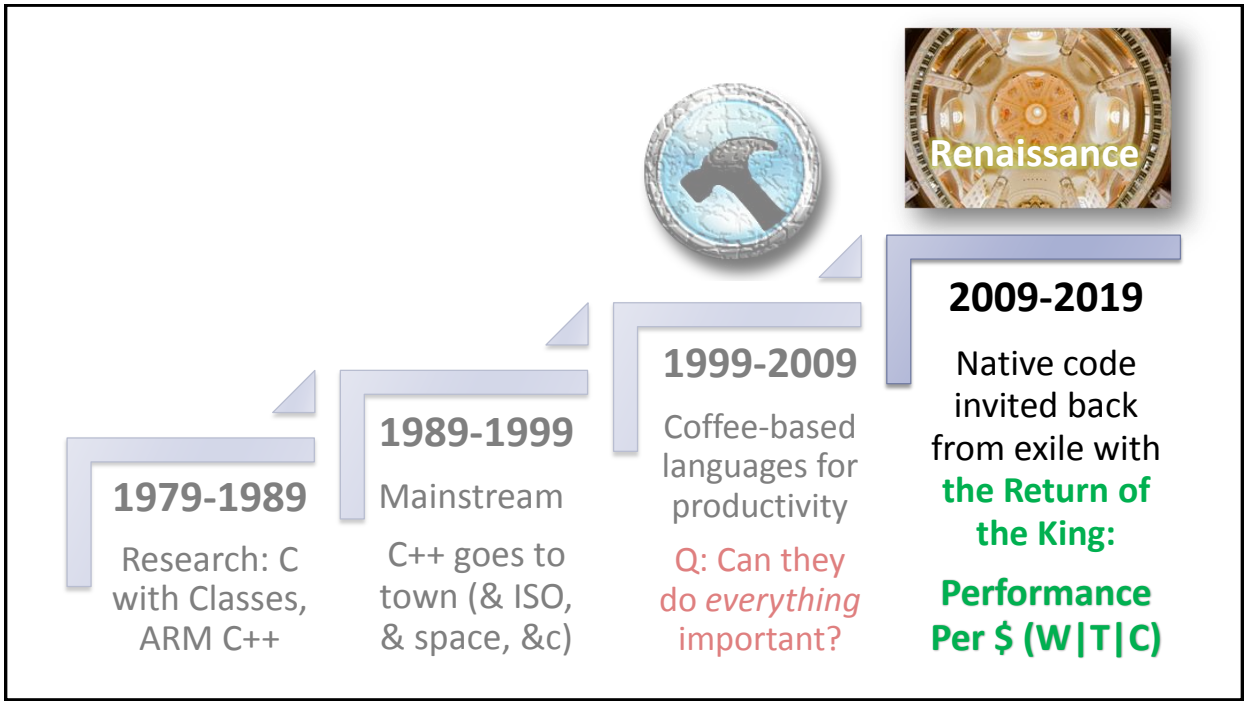
Coffee-based languages for productivity

Q: Can they do *everything* important?


English to Spanish
Grilled Sausages
French Fries
Lemonade
English to Spanish
A LA PARRILLA SALCHICHAS
FRANCÉS PATATAS FRITAS
LIMONADA

> 3. Efficiency and performance optimization will get more, not less, important. Those languages that already lend themselves to heavy optimization will find new life; those that don't will need to find ways to compete and become more efficient and optimizable. Expect long-term increased demand for performance-oriented languages and systems.

from "The Free Lunch is Over," December 2004



Reality Check

Which statement is most correct?

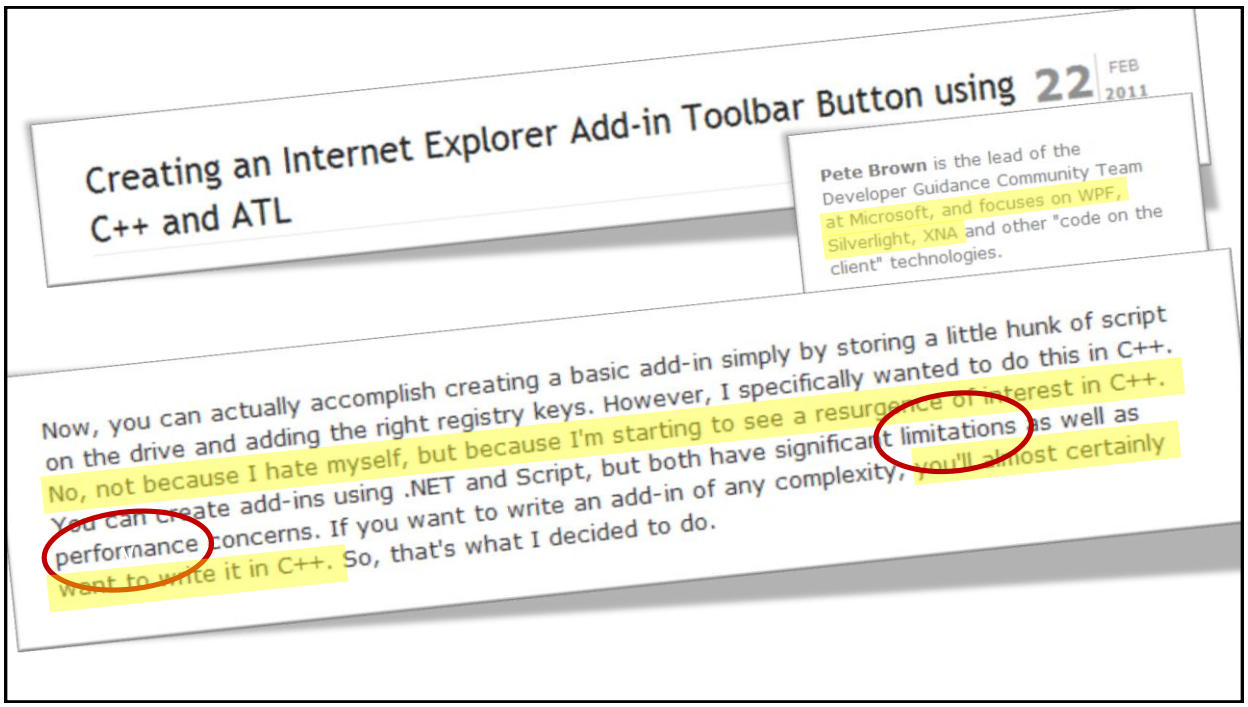
"The world is built on …

- C
- C++
- C#
- Cobol
- Fortran
- Java
- Javascript
- Perl
- Python
- Ruby

."

Where the Focus Is

	Efficiency BOT – 1999, 2009 – EOT	**Flexibility**	**Abstraction** (OO, Generics)	**Productivity** (Automatic Services, Tools)
C = efficient high-level portable code. Structs, functions.	●	●	*non-goal*	*non-goal*
C++ = C + efficient abstraction. Classes, templates.	●	●	●	*non-goal*
Java, C# = productivity. Mandatory metadata & GC, JIT compilation.	*at the expense of*	*at the expense of*	●	●



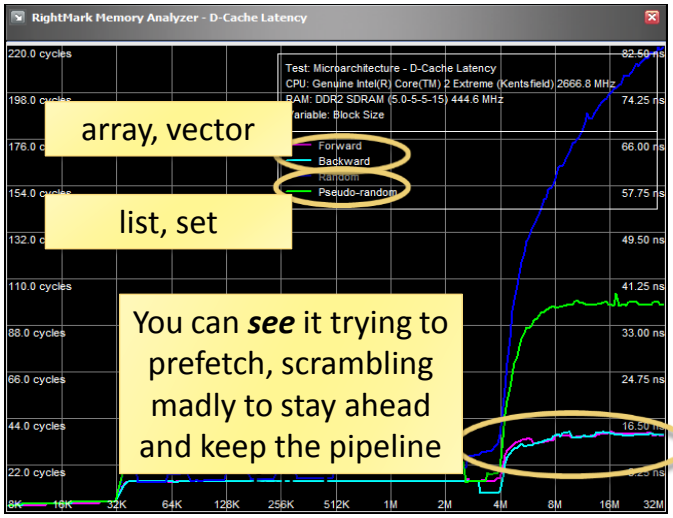
Cache-Conscious Design

EXHIBIT A FROM "MACHINE ARCHITECTURE" TALK

- **Locality is more than ever a first-order performance issue.**
- Arrange your data carefully:
  - **First: Keep data that is not used together apart,** on separate cache lines. Avoids the invisible convoying of false sharing (ping-pong).
  - **Second: Keep data that is *frequently* used together close together.** If a thread that uses A frequently also needs B, try to put them in one cache line .
  - **Third: Keep "hot" and "cold" data that is not used with the same frequency apart.** Fit "hot" data in the fewest possible cache lines and memory pages, to reduce (a) the cache footprint and cache misses, and (b) the memory footprint and virtual memory paging.

My Desktop Machine: 32K L1D$, 4M L2$

EXHIBIT B FROM "MACHINE ARCHITECTURE" TALK



- **Observations**
- Access patterns matter:
  - Linear = not bad.
  - Random = awful.
- Vectors:
  - Smaller = further left, faster.
  - Lower curves = faster.
- Lists and sets:
  - Bigger = further right, slower.
  - On higher curves = slower.

Exhibit: Size & Traversal Order

EXHIBIT B FROM "MACHINE ARCHITECTURE" TALK

- Q: Is it faster to sum an array of ints, an equivalent list of ints, or an equivalent set of ints? Which, how much so, and why?
- A1: Size matters. (Smaller = further left on the curve.)
  - All those object headers and links waste space.
- **A2: Traversal order can matter more.** (Linear = lower curve.)
  - Takes advantage of prefetching (as discussed).
  - Takes advantage of modern processors: They can potentially zoom ahead and make progress on several items in the array at the same time. Pointer-chasing = keep hitting latency.

Mobile


mobile requirements

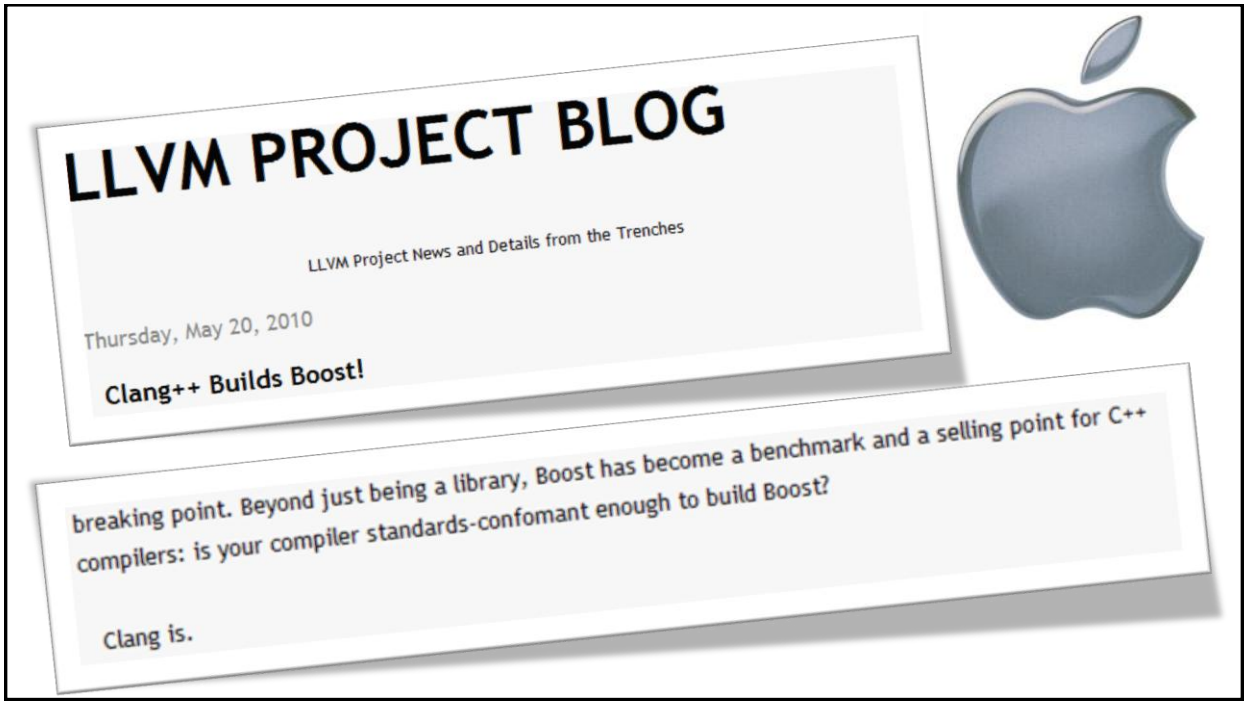
battery: lower power, longer life

size: amount of hardware you can have

bigger experiences on smaller hardware: pushing the envelope means every cycle matters

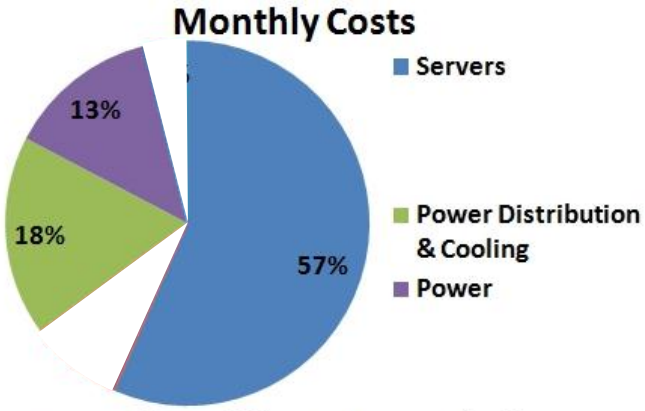


Version 1	–	Java	.NET
Version 2+	Objective-C, **C & C++** incl. C++ wrappers over Objective-C	Java, **C & C++** NDK incl. Java-free C++ apps	?


LLVM PROJECT BLOG
LLVM Project News and Details from the Trenches
Thursday, May 20, 2010
Clang++ Builds Boost!
breaking point. Beyond just being a library, Boost has become a benchmark and a selling point for C++ compilers: is your compiler standards-confomant enough to build Boost?
Clang is.


LLVM PROJECT BLOG
LLVM Project News and Details from the Trenches
Monday, May 23, 2011
C++ at Google: Here Be Dra
Google has one of the largest monolithic C++ codebases in the world. We have thousands of engineers working on millions of lines of C++ code every day. To help keep the entire thing running and all these engineers fast and productive we have had to build some unique C++ tools, centering around the Clang C++ compiler. These
Google

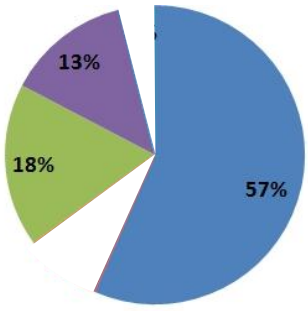
Datacenter



James Hamilton
http://perspectives.mvdirona.com/2010/09/18/OverallDataCenterCosts.aspx

Observations *[Enterprise vs. Cloud]*:

- People costs shift from top to nearly irrelevant.
- Work done/$ and work done/W are what really matters (S/W issues dominate).
- Expect high-scale service techniques to spread to enterprise.

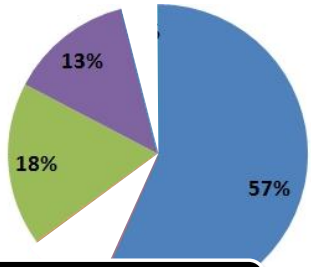


James Hamilton
VP & Distinguished Engineer, Amazon Web Services
http://www.mvdirona.com/jrh/TalksAndPapers/JamesHamilton_USENIX2009.pdf
http://perspectives.mvdirona.com/2010/09/18/OverallDataCenterCosts.aspx
http://mvdirona.com/jrh/TalksAndPapers/JamesHamilton_WhereDoesThePowerGo.pdf

Observations

Programmers & Admins **(Productivity = Coffee)**

- People costs shift from top to nearly irrelevant.
- Work done/$ and work done/W are what really matters (S/W issues dominate).
- Expect high-scale service techniques to spread to enterprise.



HW + Power = 88% **(Performance = Native)**

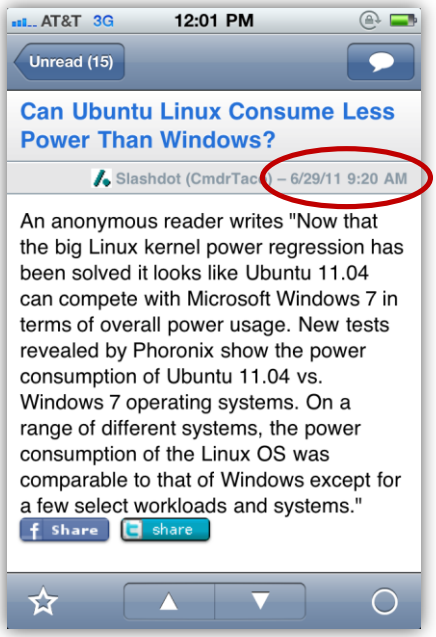
Blowback to (rest of) mainstream

James Hamilton
VP & Distinguished Engineer, Amazon Web Services
http://www.mvdirona.com/jrh/TalksAndPapers/JamesHamilton_USENIX2009.pdf
http://perspectives.mvdirona.com/2010/09/18/OverallDataCenterCosts.aspx
http://mvdirona.com/jrh/TalksAndPapers/JamesHamilton_WhereDoesThePowerGo.pdf

> My contribution to the fight against global warming is C++'s efficiency: Just think if Google had to have twice as many server farms! Each uses as much energy as a small town. And it's not just a factor of two…
>
> Efficiency is not just running fast or running bigger programs, it's also running using less resources.

Bjarne Stroustrup, June 2011

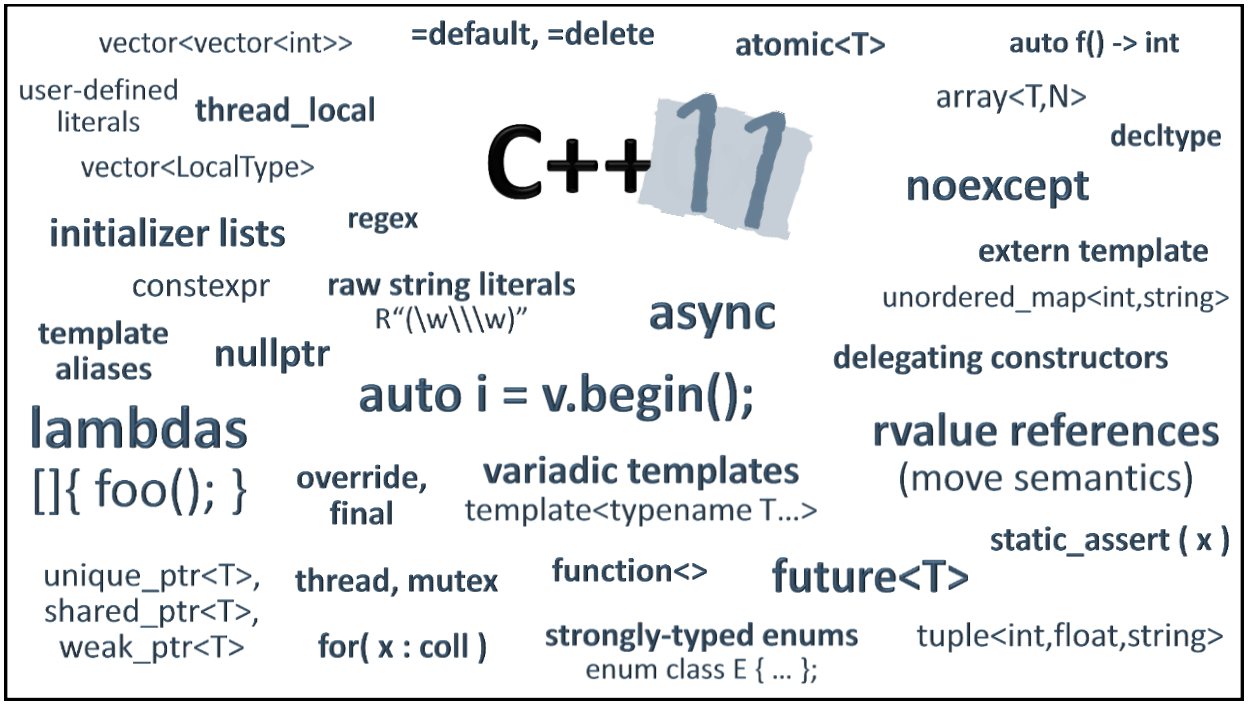
Server and Desktop

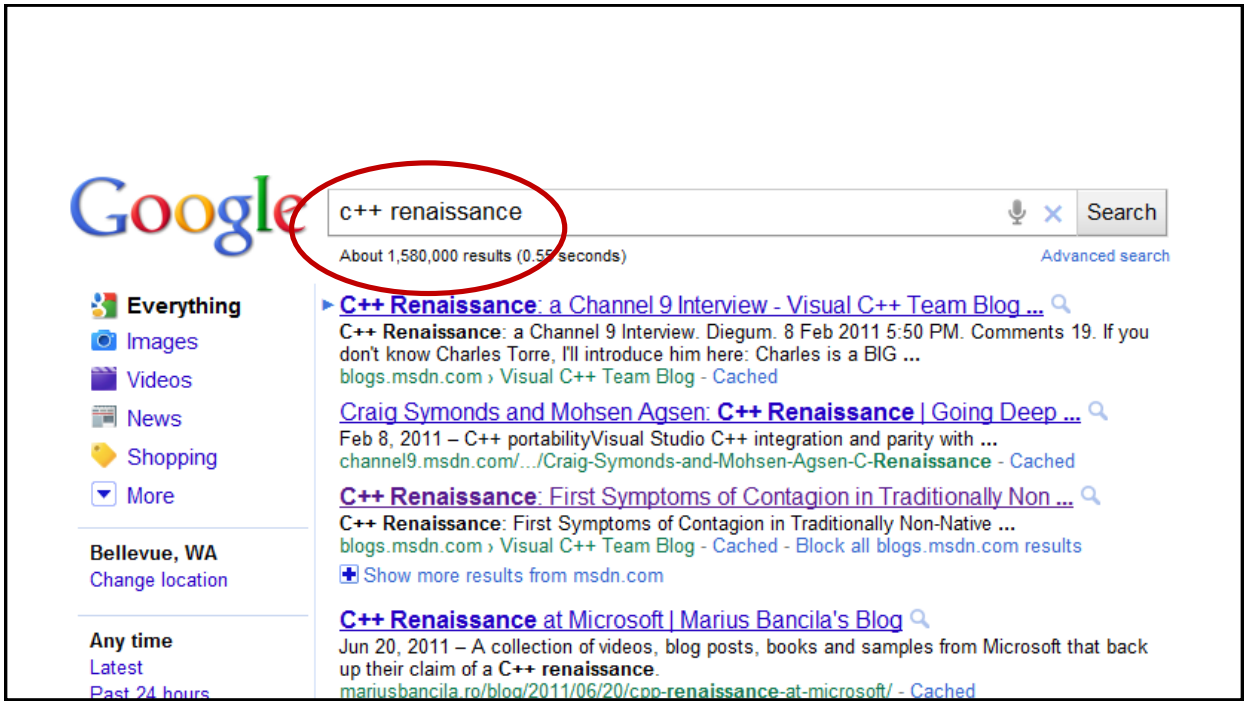


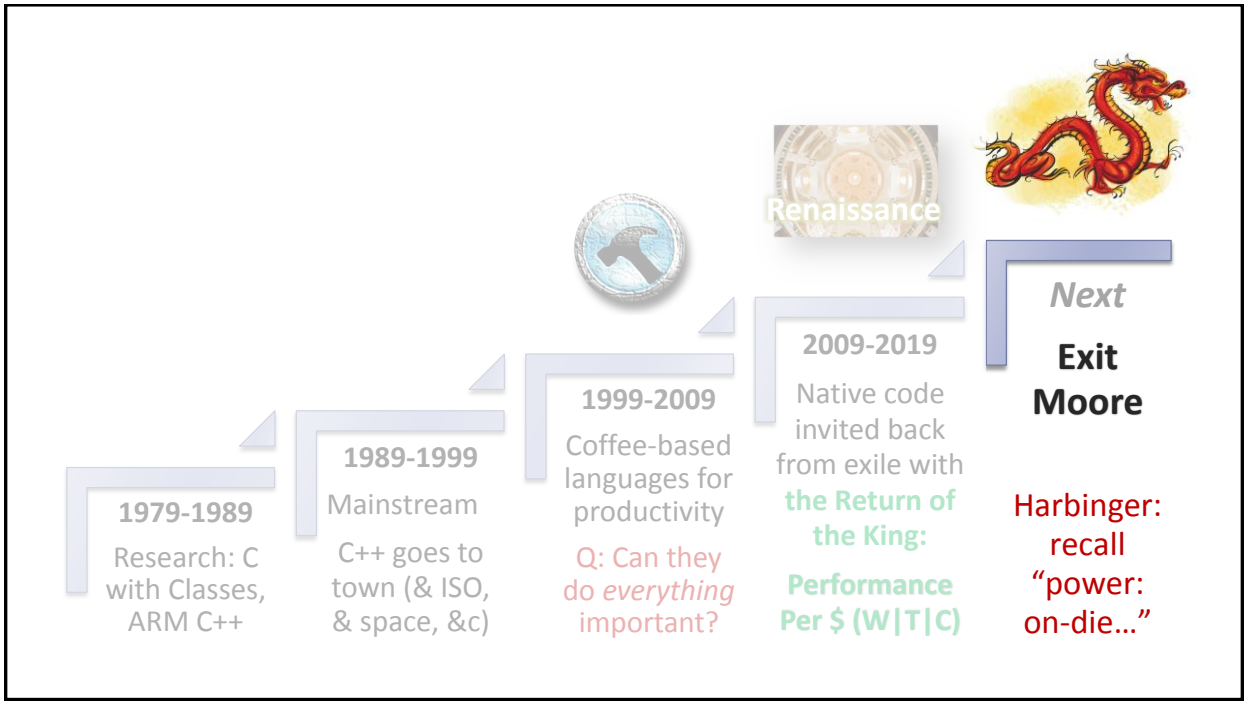
Renaissance

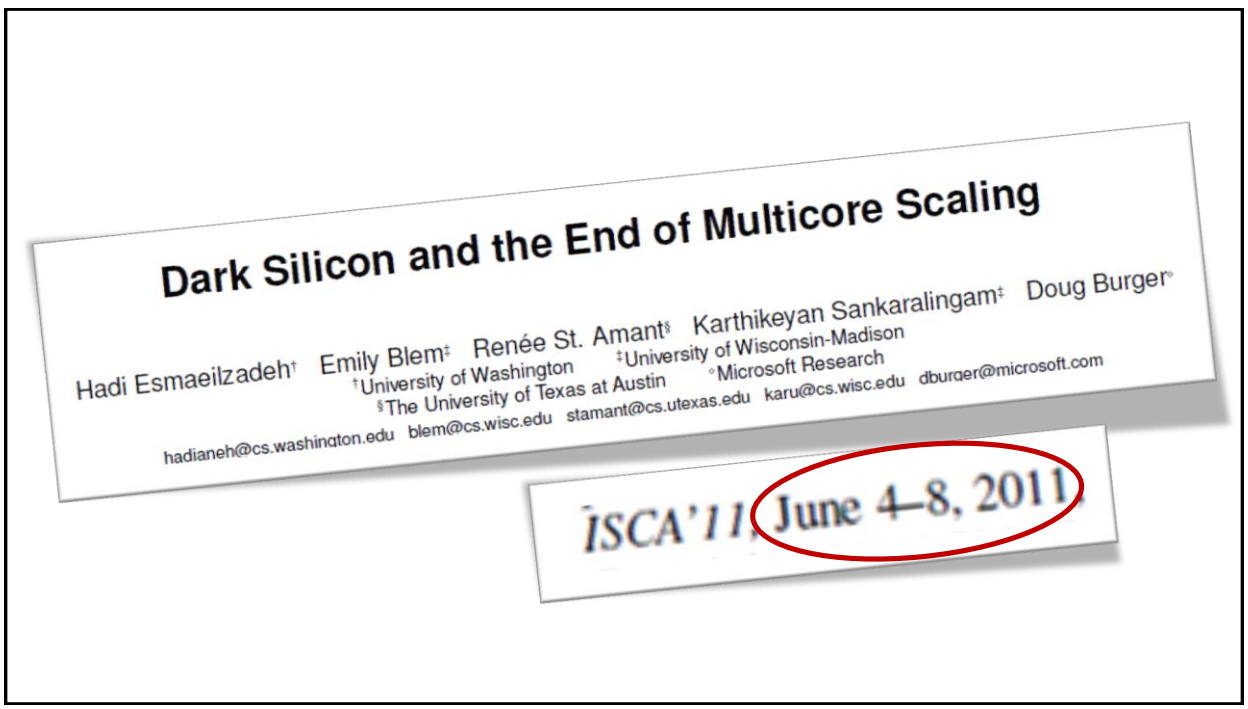
Where the Focus Is

	Efficiency (Perf/$)	**Flexibility** (Do What You Need)	**Abstraction** (OO, Generics)	**Productivity** (Automatic Services, Tools)
1970s: C				
1980s: Smalltalk, Ada, C++				
1990s: C++, Visual Basic, Delphi, Java				
2000s: .NET, Java, Objective-C, C++, C				
2010s: C++, C				


vector<vector<int>>
=default, =delete
atomic<T>
auto f() -> int
user-defined literals
thread_local
array<T,N>
decltype
vector<LocalType>
C++11
noexcept
initializer lists
regex
extern template
constexpr
raw string literals
R"(\w\\\w)"
async
unordered_map<int,string>
template aliases
nullptr
delegating constructors
auto i = v.begin();
lambdas
[]{ foo(); }
override, final
variadic templates
template<typename T...>
rvalue references
(move semantics)
static_assert (x)
unique_ptr<T>, shared_ptr<T>, weak_ptr<T>
thread, mutex
function<>
future<T>
for(x : coll)
strongly-typed enums
enum class E { ... };
tuple<int,float,string>


Google
c++ renaissance
Search
About 1,580,000 results (0.55 seconds)
Advanced search
Everything
Images
Videos
News
Shopping
More
Bellevue, WA
Change location
Any time
Latest
Past 24 hours
C++ Renaissance: a Channel 9 Interview - Visual C++ Team Blog ...
C++ Renaissance: a Channel 9 Interview. Diegum. 8 Feb 2011 5:50 PM. Comments 19. If you don't know Charles Torre, I'll introduce him here: Charles is a BIG ...
blogs.msdn.com › Visual C++ Team Blog - Cached
Craig Symonds and Mohsen Agsen: C++ Renaissance | Going Deep ...
Feb 8, 2011 – C++ portabilityVisual Studio C++ integration and parity with ...
channel9.msdn.com/.../Craig-Symonds-and-Mohsen-Agsen-C-Renaissance - Cached
C++ Renaissance: First Symptoms of Contagion in Traditionally Non ...
C++ Renaissance: First Symptoms of Contagion in Traditionally Non-Native ...
blogs.msdn.com › Visual C++ Team Blog - Cached - Block all blogs.msdn.com results
Show more results from msdn.com
C++ Renaissance at Microsoft | Marius Bancila's Blog
Jun 20, 2011 – A collection of videos, blog posts, books and samples from Microsoft that back up their claim of a C++ renaissance.
mariusbancila.ro/blog/2011/06/20/cpp-renaissance-at-microsoft/ - Cached


1979-1989
Research: C with Classes, ARM C++
1989-1999
Mainstream
C++ goes to town (& ISO, & space, &c)
1999-2009
Coffee-based languages for productivity
Q: Can they do *everything* important?
Renaissance
2009-2019
Native code invited back from exile with the Return of the King:
Performance Per $ (W|T|C)
Next
Exit Moore
Harbinger: recall "power: on-die…"


Dark Silicon and the End of Multicore Scaling
Hadi Esmaeilzadeh† Emily Blem‡ Renée St. Amant§ Karthikeyan Sankaralingam‡ Doug Burger*
†University of Washington ‡University of Wisconsin-Madison
§The University of Texas at Austin *Microsoft Research
hadianeh@cs.washington.edu blem@cs.wisc.edu stamant@cs.utexas.edu karu@cs.wisc.edu dburger@microsoft.com
ISCA'11 June 4–8, 2011

Why C++ ?

power: driver at all scales: on-die, mobile, desktop, datacenter

size: limits on processor resources: desktop, mobile

experiences: bigger experiences on smaller hardware; pushing envelope means every cycle matters

Why, C++ !


C++ and Beyond 2011, Banff
Welcome!