Fast campus

CS SCHOOL

앞으로 C/C++에서의 pointer와 string을 배울 겁니다.

파이썬에서는 string을 쓰는 데 많이 고려할 필요도 없었고
아주 쉽게 배웠었지요~

하지만 C/C++에서의 string은 매우 어렵습니다.

많은 분들이 그냥 쉬운 걸 배우면 되지 않냐고 물어보시는데요.

저는 그렇게 생각하지 않습니다. 웹 서비스 성능이나
텍스트 데이터 처리에서도 string을 제대로 아는 것은 매우 중요합니다.

오늘 아침, 나프다(나는 프로그래머다)를 보다가 좋은 컨퍼런스 영상을
발견했습니다.

내용도 어려운데 한글 자막이 없어서 제가 2/3 정도 부분만 번역하여 내용을
정리했습니다.

앞으로 포인터와 string을 배운 후 다시 보신다면 큰 도움이 될 것입니다.

C++에서의 string

<string>이라는 STL(standard template library)

```
#include <iostream>
#include <string>
using namespace std;

int main(void)
{
    char * str = "hello"; 우리가 배울 C-style

    string st1(str);
    string st2("world");
    string st3 = st1 + " " + st2;//문자열 + 연산자 가능
    string st4("hello", 4);//문자열 중 4개 문자로 만듦
    string st5(st1.begin(), st1.end());
```

Fast campus

The strange details of std::string at Facebook

Nicholas Ormrod

Software Engineer

CppCon 2016

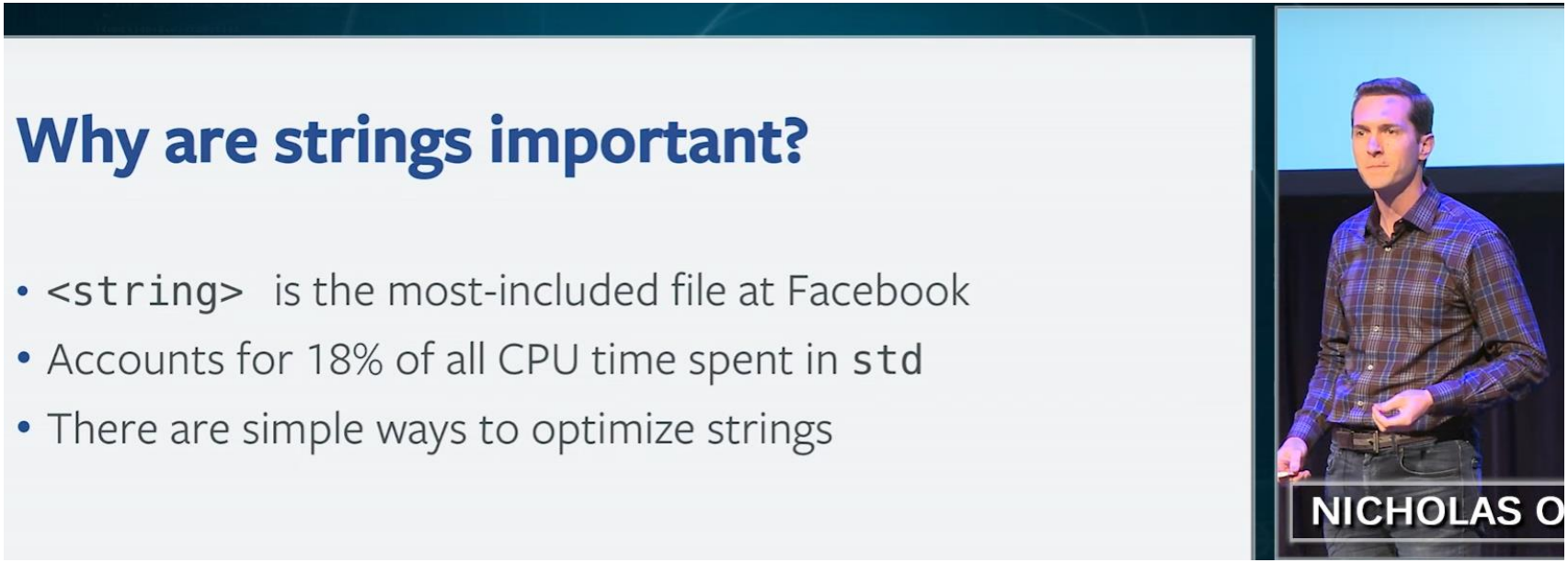
Questions I want answers for

- What is the most efficient string implementation?

니콜라스(발표자)는 무엇이 페이스북에서 가장 효율적인 String인지, 여러분에게는 무엇이 가장 효율적인지 자신은 잘 모른다고 하면서 세션을 시작합니다.

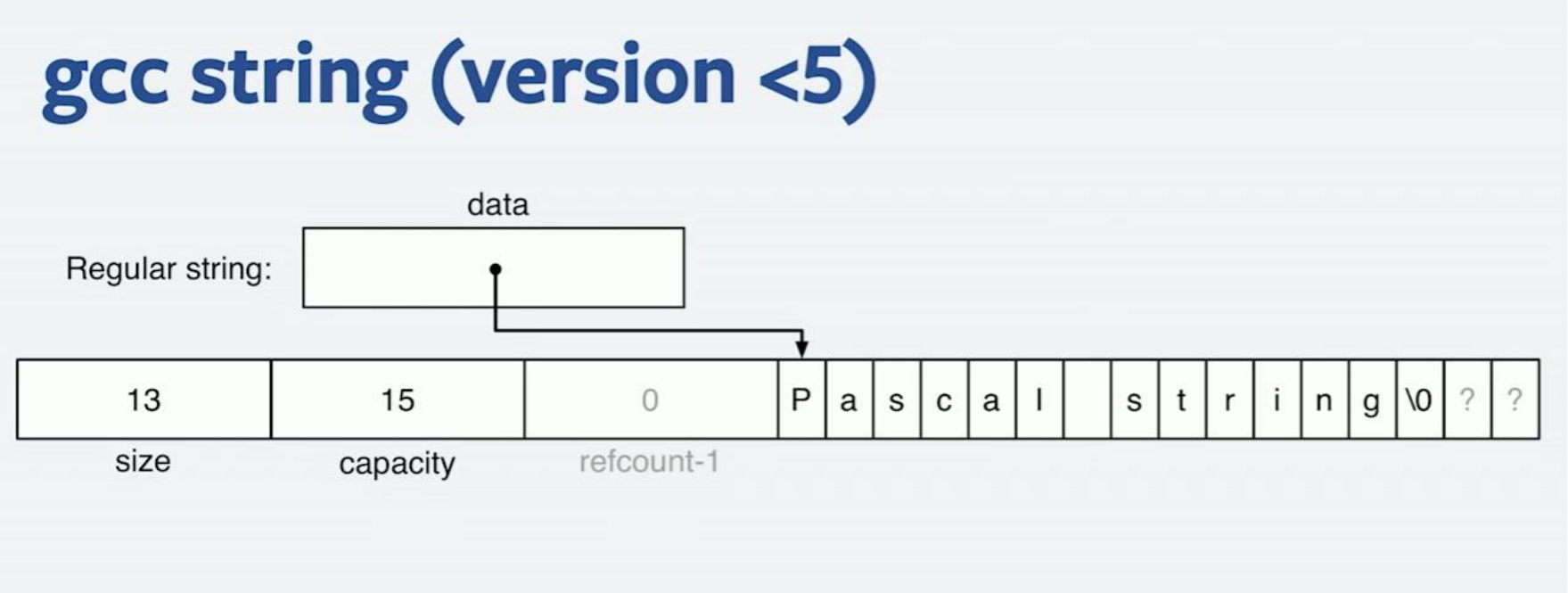
파이썬만 배운 이 시점에서는 이 발표자가 무슨 말을 하는지 알 수 없습니다. 강연을 조금만 더 들어보도록 하죠!

"The most important thing is.....
Strings are important!"



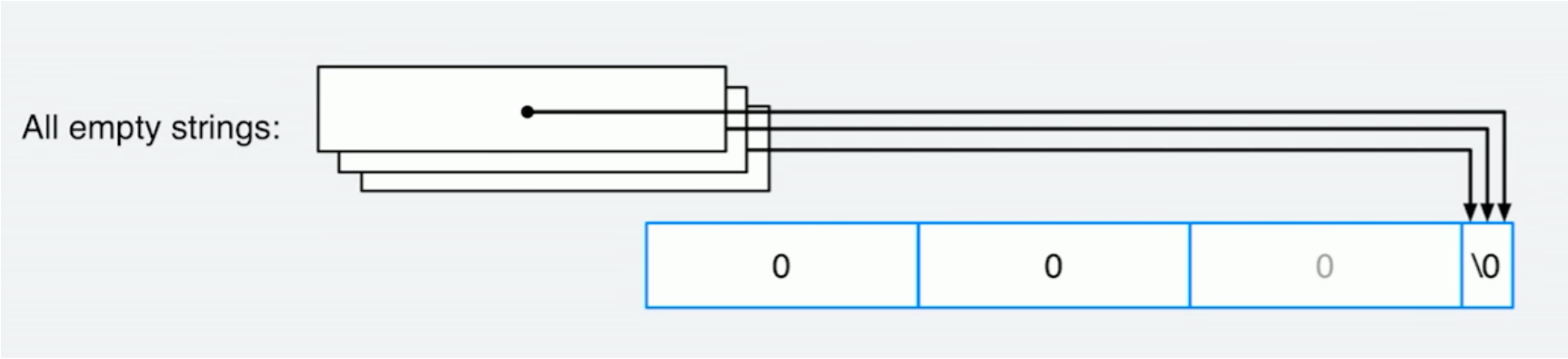
Software engineer at facebook
NICHOLAS

자그마치 6년 전 gcc(linux 계열이 사용하는 C/C++ 컴파일러)는
아래와 같은 구조로 짜여져 있었다고 합니다.



실제 string data는 Heap 영역에 할당된다고 합니다.

모든 string은 비어 있더라도 반드시 1byte를 가지고 있다.
왜냐하면 '₩0' 문자가 있어야 하기 때문



Global empty string array

Data 영역에 미리 만들어 놓아
모든 empty string이 가리키도록 합니다.

"There are lots of Facebook programs
that are bottlenecked on string codes"

문제는 malloc()이거나
여기 저기 페이지를 넘나들며 string 데이터에 접근할 때
발생했다고 합니다.

"strings are slow!!"

String은 원래 느리다고 강조를 하는군요!

fbstring

@author Andrei Alexandrescu

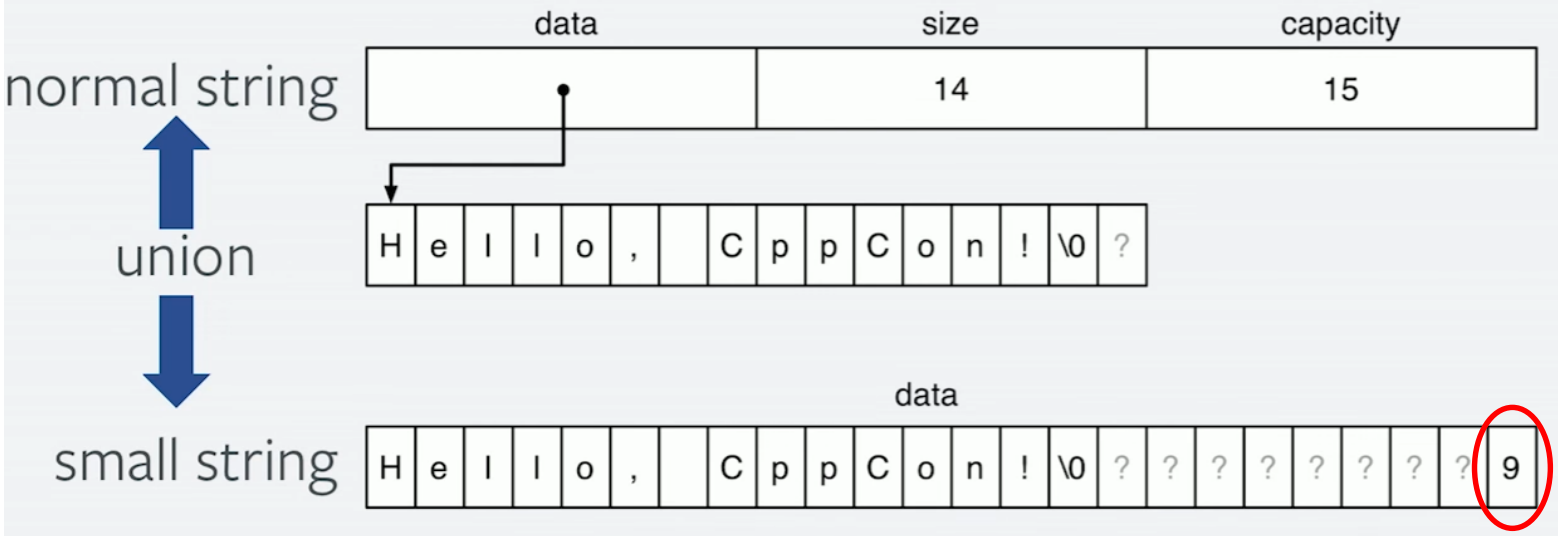
@ 페이스북 엔지니어들의 목표

String이 충분히 작다면

- malloc() 함수를 최대한 쓰지 않고
- radom하게 할당되는 heap이 아니라
- stack에 할당하는 것

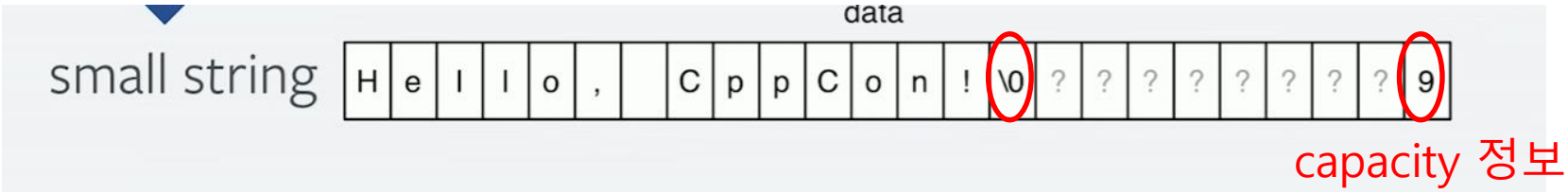
fbstring

@author Andrei Alexandrescu



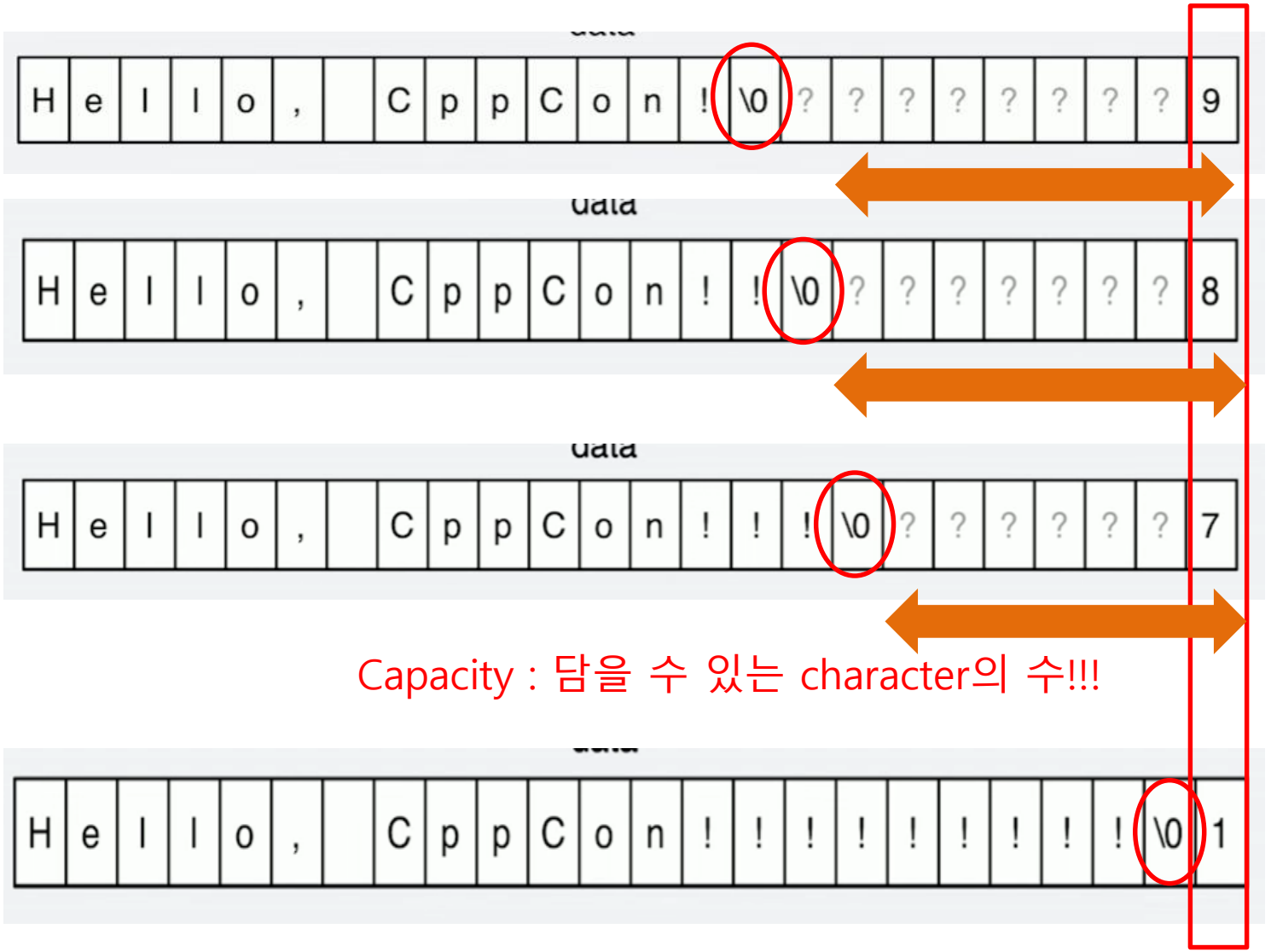
Stack에 24 bytes의 구조로 다시 만든 string

이 수의 정체는?

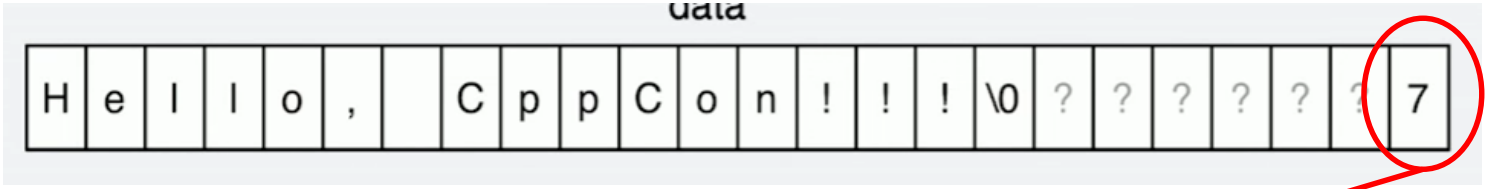


24 bytes 구조 중 몇 byte를 실제 데이터로 쓸 수 있을까요?

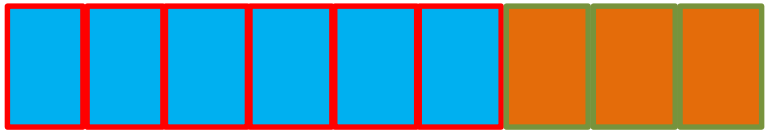
- '₩0' 문자 하나와 capacity나 flag 정보를 담을 1byte를 빼고
- 아마 22 bytes가 최대로 저장할 수 있는 실제 데이터일 듯.....



'₩0'이 0이므로 0은 두 가지 정보를 모두 담을 수 있다!!!!!



그럼 small string인지 normal string인지 구분하는
Flag 정보는 어디에??



1 bit면 flag 정보를 담을 수 있다!

Small strin은 23 bytes 이므로 $2^5 = 32$, 즉 5 bit면 충분히 표현 가능!

자, 이제 gcc 내장 string 구조체와 성능을 비교해봅시다!

Fast campus

Performance of fbstring

gcc_string.size()

```
movq    (%rdi), %rax
movq    -24(%rax), %rax
```

어디서 많이 본.....
지난 시간에 공부한 어셈블리어?!?!

fbstring.size()

```
movabsq $-4611686018427387904, %rax
testq   %rax, 16(%rdi)
je      .L7

movq    8(%rdi), %rax
ret
.L7:
movsbq  23(%rdi), %rdx
movl    $23, %eax
subq    %rdx, %rax
ret
```

Gcc_string.size() 코드는 두 줄이고 fbstring.size()는 여러 줄이니.....
gcc가 더 빠르겠네요?!?!

Performance of fbstring

gcc_string.size()

```
movq    (%rdi), %rax
movq    -24(%rax), %rax
```

1.6ns

fbstring.size()

```
movabsq $-4611686018427387904, %rax
testq   %rax, 16(%rdi)
je      .L7
movq    8(%rdi), %rax
ret
.L7:
movsbq  23(%rdi), %rdx
movl    $23, %eax
subq    %rdx, %rax
ret
```

분기문.....if까지 들어가 있네요.....

is_small

subtract A B

0.9ns

그런데 속도는 fbstring이 거의 반절 가까이 빠르다?!?!

페이스북 어셈블리어 9줄이
gcc의 어셈블리어 2줄 보다 빠른
비밀은 ……..

Memory layout of your program 때문이랍니다.

gcc의 string 데이터는 heap에 할당되고
Size() 함수를 호출할 때 마다 다른 page로 가서 size 데이터를
가져와야 하므로 느릴 수 밖에 없다는 것이죠!
(이 부분은 가상 메모리에 대해 공부하면 조금은 명확해 집니다)

Small string의 경우 contiguous 하게 stack에 할당되는 fbstring보다
Gcc의 string이 page miss나 L1 cache miss가 3배나 높다고 하네요.

std::string replacement

- We replaced `std::string`'s implementation with fbstring's
- `std::string` and `folly::fbstring` now have the same implementation, but are still different types

1% performance win

It sounds small but IT IS A HUGE NUMBER!!!!!!!

1%의 성능 향상은 작아 보일지 몰라도 이 수치는 굉장한 겁니다!
상상해보세요 페이스북 서버에서 돌아가는 모든 C++ 프로그램
그 중 string 데이터에서 1%의 성능 향상입니다.....
어마 무시한 수치이지요!!!