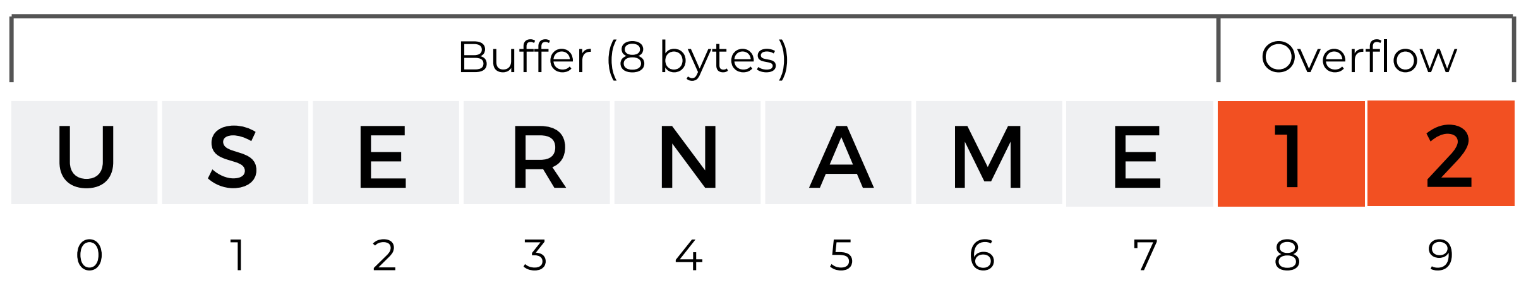
CHECKED C FOR MEMORY SAFETY

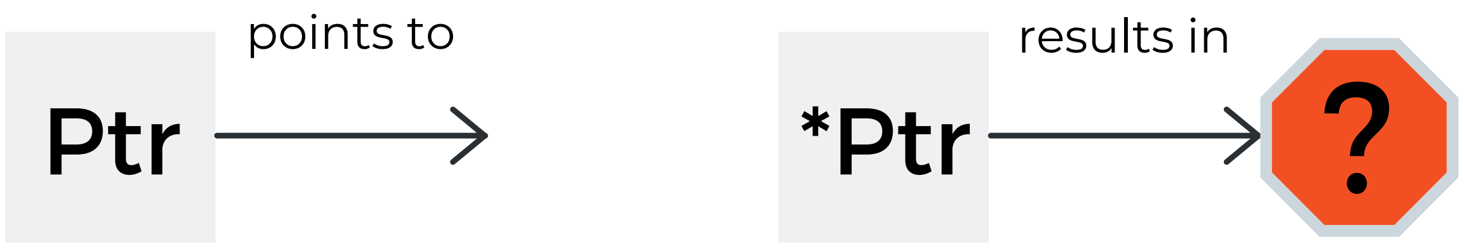
Mandeep Singh Grang . Microsoft, Redmond, WA

Two Memory Safety Hazards

1. Buffer overflows are a security threat.



2. Null pointer dereference results in undefined behavior.



Compiler Implementation

- Checked C implemented as an extension to the LLVM/Clang compiler.
- Extended the C grammar to support bounds declarations and the new types _Ptr<T>, etc.
- Added checked headers which ascribe bounds-safe interfaces to standard library functions.
- Extended Clang's AST and Type system to support the new checked types.
- Extended Clang's checker to check sanity of bounds declarations.
- Total lines of compiler code: 20 K

Checked C Resources

Checked C Code Repository
https://github.com/Microsoft/checkedc-clang

Checked C Language Specification
https://github.com/Microsoft/checkedc/releases

Checked C SecDev 2018 Paper
https://www.microsoft.com/en-us/research/publication/checkedc-making-c-safe-by-extension

What is Checked C?

- Extension to C designed to support spatial safety.
- Supports incremental porting from legacy C.
- Adds new pointer and array types that are bounds-checked.

`_Ptr<T>`

- For pointers to singleton objects.
- Runtime check ensures pointer dereference is non-null.

```
_Ptr<int> p = 0;
void foo(_Ptr<int> p);
_Ptr<char> bar();
```

`_Array_ptr<T>`
`T _Checked[]`

- For arrays and pointers involved in pointer arithmetic.
- Runtime check ensures pointer access is within bounds.

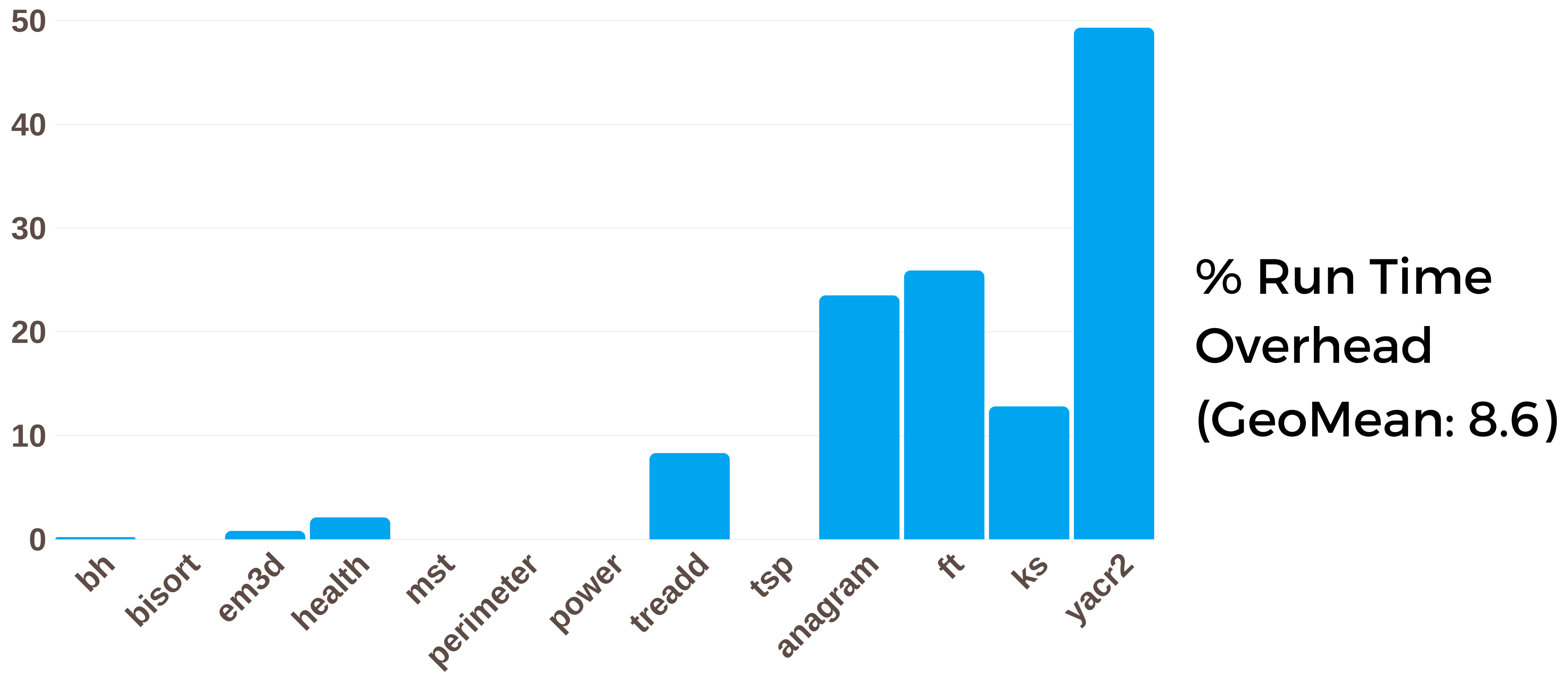
```
_Array_ptr<char> c : count(3) = "abc";
_Array_ptr<int> i : bounds(i, i+3) = {1, 2, 3};
float f _Checked[2][2] = {{1.0, 2.0}, {3.0, 4.0}};
```

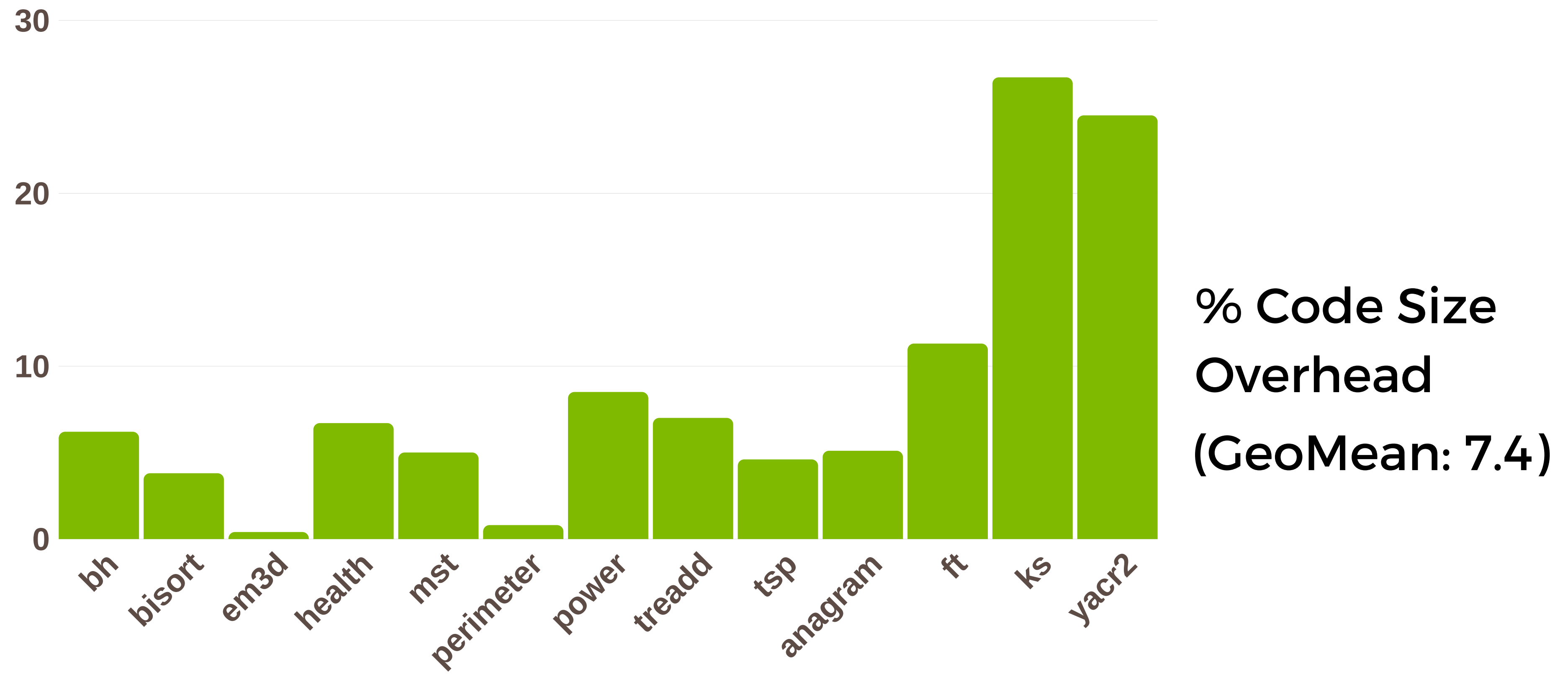
`_Nt_array_ptr<T>`
`T _Nt_checked[]`

- Variant of _Array_ptr<T> for null-terminated arrays.
- Size of the array includes the null terminator element.
- Runtime check ensures null terminator is not overwritten.

```
_Nt_array_ptr<char> s : count(3) = "abc";
char s _Nt_checked[4] = "abc";
```

Experimental Evaluation of Checked C





Note: Average Benchmark LOC Modified: 17.5 %

LLVM Developers' Meeting 2019, San Jose, CA