Protokol o provedeném profilingu

Použitý profiler:

Byl použit vestavěný profiler v prostředí Node.js. Pro získání zpracovávaného výstupu byly použity následující příkazy:

```
node –prof std_dev.js < data_N.txt
node –prof-process v8.log > results_N.txt
```

Vstup a výstup profileru:

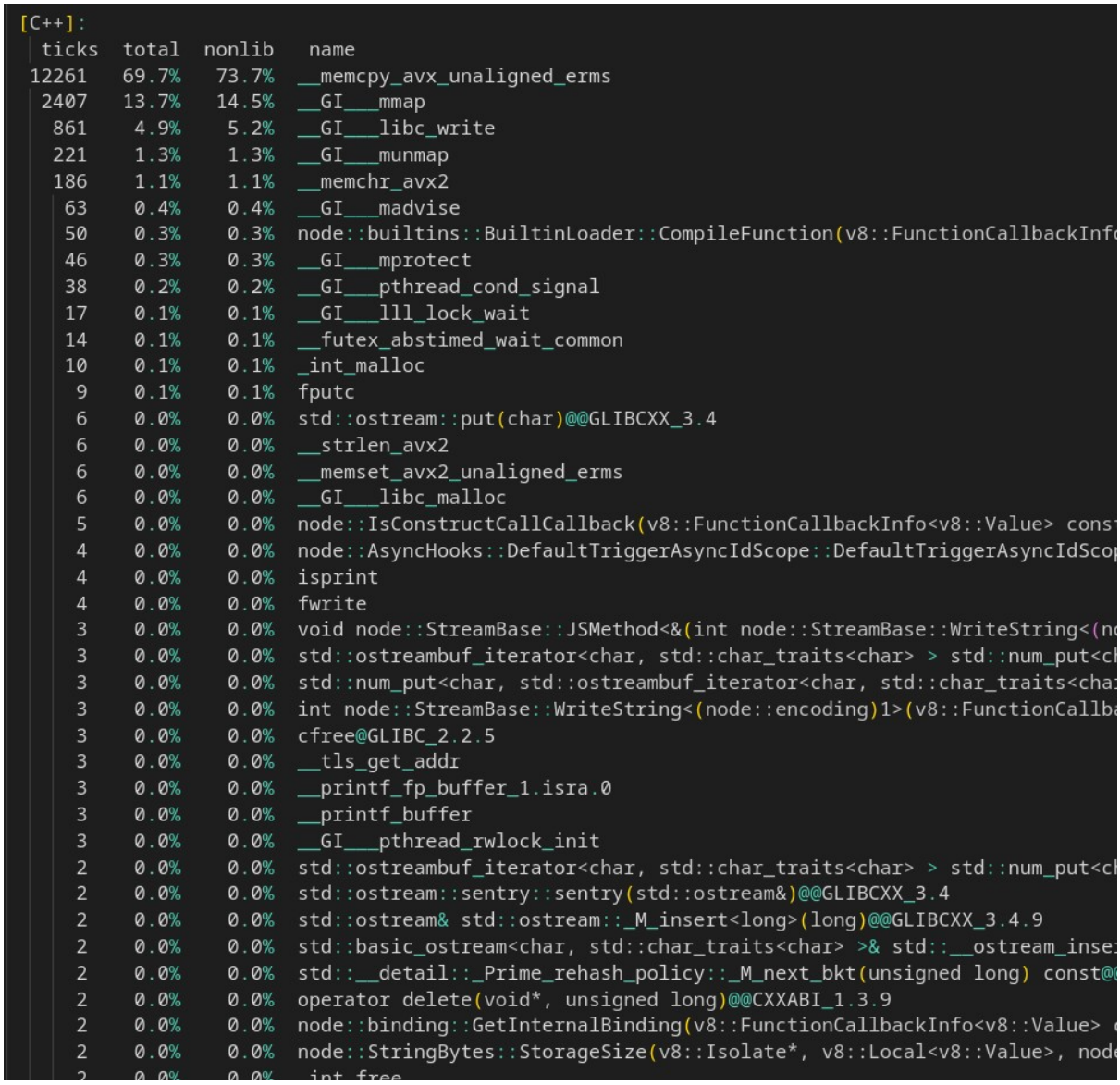
Program pro výpočet směrodatné odchylky jsme profilovali se vstupy o velikostech 10, 1000 a 50000 čísel (v souborech data_N.txt ve složce profiling). V souborech results_N.txt jsou uloženy výsledky jednotlivých běhů profilingu.

Pro ilustraci je část obsahu souboru results_50000.txt přiložena i zde:

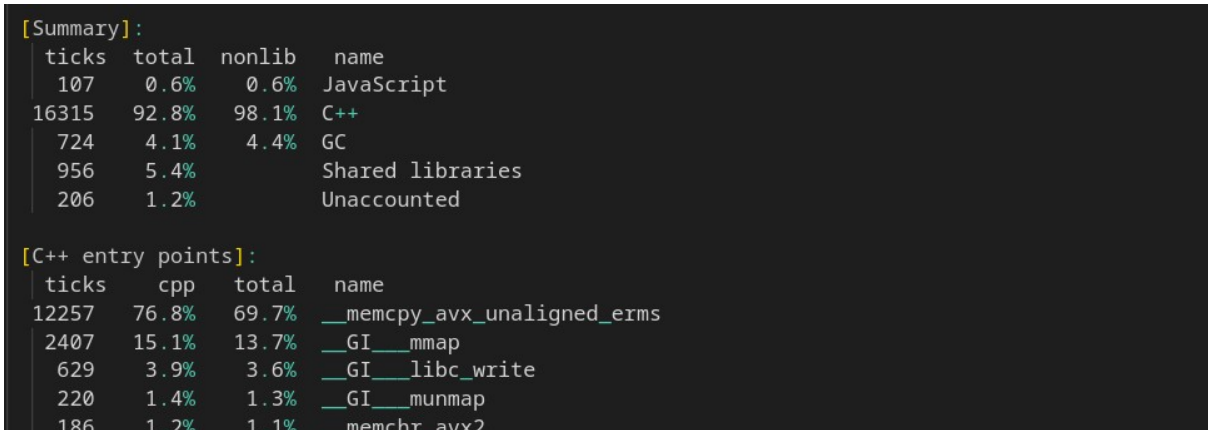
```
Statistical profiling result from isolate-0x5581c95c73d0-308566-v8.log, (17584 ticks, 206 unacc

[Shared libraries]:
  ticks  total  nonlib   name
    946    5.4%          /usr/lib64/libnode.so.115
      5    0.0%          /usr/lib64/libstdc++.so.6.0.32
      4    0.0%          [vdso]
      1    0.0%          /usr/lib64/libc.so.6

[JavaScript]:
  ticks  total  nonlib   name
     15    0.1%    0.1%  JS: *getCursorPos node:internal/readline/interface:1022:15
     12    0.1%    0.1%  JS: *_solveComplexEquations /home/otakarkoci/git/ivs_project_kvm_switc
      9    0.1%    0.1%  JS: *writeOrBuffer node:internal/streams/writable:540:23
      9    0.1%    0.1%  JS: *onkeypress node:internal/readline/interface:263:22
      7    0.0%    0.0%  RegExp: \r?\n|\r(?!\n)
      7    0.0%    0.0%  JS: *emitKeys node:internal/readline/utils:86:19
      6    0.0%    0.0%  RegExp: [\u001B\u009B][[\]()#;?]*(?:(?:(?:(?:;[-a-zA-Z\d\/#&.:=?%@~_]+
      5    0.0%    0.0%  RegExp: (?<!\d+\.)\d+!
      4    0.0%    0.0%  RegExp: ^[0-9A-Za-z]$
      4    0.0%    0.0%  JS: *<anonymous> node:internal/readline/interface:1056:14
      3    0.0%    0.0%  RegExp: [+\-*/()]
      3    0.0%    0.0%  RegExp: -?\d+(?:\.\d+)?(?:e[+-]\d+)?
      3    0.0%    0.0%  JS: *onData node:internal/readline/emitKeypressEvents:46:18
      2    0.0%    0.0%  RegExp: pow\(-?\d+(?:\.\d+)?(?:e[+-]\d+)?,-?\d+(?:\.\d+)?(?:e[+-]\d+)?
      2    0.0%    0.0%  RegExp: ^[A-Z]$
      2    0.0%    0.0%  RegExp: [\(\{\[][-+]?\d+(?:\.\d+)?(?:e[+-]\d+)?(((([/*][+-])|[-+*/])\d+
      2    0.0%    0.0%  RegExp: ([/*][+-])|[-+*/]
      1    0.0%    0.0%  RegExp: \s+
      1    0.0%    0.0%  RegExp: \d+(?:\.\d+)?(?:e[+-]\d+)?
      1    0.0%    0.0%  JS: ^writeOrBuffer node:internal/streams/writable:540:23
      1    0.0%    0.0%  JS: ^clearLine node:internal/readline/interface:874:12
      1    0.0%    0.0%  JS: ^_solveOneEquationBasicMath /home/otakarkoci/git/ivs_project_kvm_s
      1    0.0%    0.0%  JS: ^_findLowestBracket /home/otakarkoci/git/ivs_project_kvm_switchers
      1    0.0%    0.0%  JS: ^<anonymous> node:internal/readline/interface:440:16
      1    0.0%    0.0%  JS: *_solveOneEquationBasicMath /home/otakarkoci/git/ivs_project_kvm_s
      1    0.0%    0.0%  JS: *_findLowestBracket /home/otakarkoci/git/ivs_project_kvm_switchers
      1    0.0%    0.0%  JS: *<anonymous> node:internal/readline/interface:432:19
      1    0.0%    0.0%  JS: *<anonymous> node:internal/readline/interface:1030:16
      1    0.0%    0.0%  JS: *<anonymous> /home/otakarkoci/git/ivs_project_kvm_switchers/src/js
```



```
[C++]:
 ticks  total  nonlib   name
12261   69.7%   73.7%  __memcpy_avx_unaligned_erms
 2407   13.7%   14.5%  __GI___mmap
  861    4.9%    5.2%  __GI___libc_write
  221    1.3%    1.3%  __GI___munmap
  186    1.1%    1.1%  __memchr_avx2
   63    0.4%    0.4%  __GI___madvise
   50    0.3%    0.3%  node::builtins::BuiltinLoader::CompileFunction(v8::FunctionCallbackInf
   46    0.3%    0.3%  __GI___mprotect
   38    0.2%    0.2%  __GI___pthread_cond_signal
   17    0.1%    0.1%  __GI___lll_lock_wait
   14    0.1%    0.1%  __futex_abstimed_wait_common
   10    0.1%    0.1%  _int_malloc
    9    0.1%    0.1%  fputc
    6    0.0%    0.0%  std::ostream::put(char)@@GLIBCXX_3.4
    6    0.0%    0.0%  __strlen_avx2
    6    0.0%    0.0%  __memset_avx2_unaligned_erms
    6    0.0%    0.0%  __GI___libc_malloc
    5    0.0%    0.0%  node::IsConstructCallCallback(v8::FunctionCallbackInfo<v8::Value> cons
    4    0.0%    0.0%  node::AsyncHooks::DefaultTriggerAsyncIdScope::DefaultTriggerAsyncIdScop
    4    0.0%    0.0%  isprint
    4    0.0%    0.0%  fwrite
    3    0.0%    0.0%  void node::StreamBase::JSMethod<&(int node::StreamBase::WriteString<(n
    3    0.0%    0.0%  std::ostreambuf_iterator<char, std::char_traits<char> > std::num_put<ch
    3    0.0%    0.0%  std::num_put<char, std::ostreambuf_iterator<char, std::char_traits<cha
    3    0.0%    0.0%  int node::StreamBase::WriteString<(node::encoding)1>(v8::FunctionCallba
    3    0.0%    0.0%  cfree@GLIBC_2.2.5
    3    0.0%    0.0%  __tls_get_addr
    3    0.0%    0.0%  __printf_fp_buffer_1.isra.0
    3    0.0%    0.0%  __printf_buffer
    3    0.0%    0.0%  __GI___pthread_rwlock_init
    2    0.0%    0.0%  std::ostreambuf_iterator<char, std::char_traits<char> > std::num_put<ch
    2    0.0%    0.0%  std::ostream::sentry::sentry(std::ostream&)@@GLIBCXX_3.4
    2    0.0%    0.0%  std::ostream& std::ostream::_M_insert<long>(long)@@GLIBCXX_3.4.9
    2    0.0%    0.0%  std::basic_ostream<char, std::char_traits<char> >& std::__ostream_inse
    2    0.0%    0.0%  std::__detail::_Prime_rehash_policy::_M_next_bkt(unsigned long) const@@
    2    0.0%    0.0%  operator delete(void*, unsigned long)@@CXXABI_1.3.9
    2    0.0%    0.0%  node::binding::GetInternalBinding(v8::FunctionCallbackInfo<v8::Value> 
    2    0.0%    0.0%  node::StringBytes::StorageSize(v8::Isolate*, v8::Local<v8::Value>, node
    2    0.0%    0.0%  _int_free
```



```
[Summary]:
 ticks  total  nonlib   name
   107   0.6%    0.6%  JavaScript
 16315  92.8%   98.1%  C++
   724   4.1%    4.4%  GC
   956   5.4%          Shared libraries
   206   1.2%          Unaccounted

[C++ entry points]:
 ticks    cpp   total   name
12257   76.8%   69.7%  __memcpy_avx_unaligned_erms
 2407   15.1%   13.7%  __GI___mmap
  629    3.9%    3.6%  __GI___libc_write
  220    1.4%    1.3%  __GI___munmap
  186    1.2%    1.1%  __memchr_avx2
```

Z výsledků získaných námi vybraným profilerem je patrné, že pro získání jemných údajů o volaní jednotlivých JavaScriptových funkcí naší knihovny se příliš nehodil. Ve změti volání vestavěných funkcí prostředí Node.js a

implementačních funkcích interpretu JavaScriptu se volání jednotlivých funkcí matematické knihovny ztrácí a jedná se spíše o profilling prostředí Node.js samotného.

Nejvíce času je alokováno na provádění úkonů spojených s Node.js procesy. Při profilování se vstupy o délce 10 a 1000 čísel se JavaScript ani nezobrazil ve výsledcích. Při profilování na 50 000 vstupních číslech se JavaScript již ukázal. Je možné vidět, že část času zabralo inicializování regulárních výrazů a jejich používání a také metoda _solveComplexEquations. I když by mělo být zmíněno že hodnoty pohybují se okolo 0.1 % pravděpodbně nebudou příliš průkazné

Možnosti optimalizace:
Ze získaných dat nelze vyvodit jasné závěry ohledně toho, co se nejvíce vyplatí optimalizovat. Samozřejmě ale můžeme předpokládat, že zjednodušení a úpravy funkcí a struktury matematické knihovny by se projevily zjednodušením nutné práce interpretu JavaScriptu samotného. Tudíž by mohlo být vhodné například upravit a zjednodušit funkci _solveComplexEquations, či funkce pro řešení základních operací a nebo nejlépe použít vhodnou strukturu pro reprezentaci řešené rovnice jako takové, což by mohlo oproti aktualnímu neustálému textovému nahrazování podčástí za mezivýsledky výrazně urychlit výpočty.

Závěr:
Námi zvolený profiler bohužel příliš neosvětlil dění v naší matematické knihovně. Zkoušeli jsme i další řešení. Problémem ale je, že v případě JavaScriptu je většina profilerů orientovaná přímo na prohlížeče a jiné případy použití, které se našim potřebám provádět profiling automatizovaně v prostředí Node.js a v konzoli příliš nepřibližují. Pár možností na vylepšení matematické knihovny jsme sepsali výše, i když z výše zmíněných důvodů příliš nevyplývají z profilingu samotného. Na závěr by bylo vhodné řici, že se obecně JavaScript pro podobné potřeby, jako jsou matematické knihovny, příliš nehodí.