Outline

This presentation covers the following topics

- Session 1 – Cilk and the Lucata API
  - Basic programming
  - Data distribution
- Session 2 – Lucata Workflow
  - X86 Debugging
  - Simulation
  - Hardware
- **Session 3 – Measuring Performance**
  - ***Timing Hooks***
  - ***Profiling***
- Session 4 – Coding Optimizations
  - Machine-specific coding
  - Parallel computation
- Section 5
  - Advanced topics

Slides originally developed by Janice McMahon, Lucata Corporation

Measuring Performance

Timing hooks in C/C++ utilities, profiling

Functions for Timing Hooks

- Mark beginning of region

```
void hooks_region_begin(const char *name);
```

- Mark end of region

```
double hooks_region_end(const char *name);
```

- Set current region

```
void hooks_set_active_region(const char *name);
```

- Record value for region

```
void hooks_set_attr_T(const char *key, <type> value);
```

  - T = u64: record unsigned integer
  - T = i64: record signed integer
  - T = f64: record double precision
  - T = str: records string

Timing Hooks: example

```
// Initialization section, will not be timed
long n = 1024;
long * x = mw_malloc1dlong(n);
for (long i = 0; i < n; ++i) { x[i] = i; }
hooks_set_attr_i64("N", n);
// Region of interest, will be timed
hooks_region_begin("sum");
long sum = 0;
for (long i = 0; i < n; ++i) { sum += x[i]; }
double time_ms = hooks_region_end();
```

Bounded timing region; runs in timed mode (i.e., with starttiming()) for that region

Uses CLOCK() intrinsic; requires environment variables for machine parameters

Additional values can be output for the region as directed by programmer

Only one active region at a time; if none specified, first region encountered is active

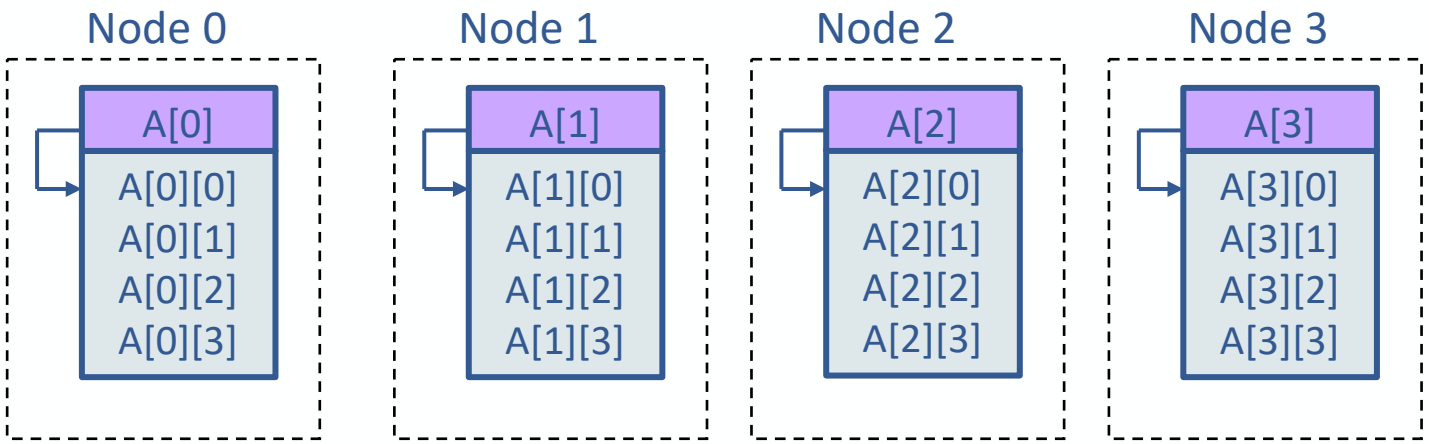
Timing output in JSON format for each invocation of region:

```
{"region_name": "sum", "time_ms": 3.14, "ticks": 1234567, "N": 1024}
```

Example: Chunked Array of Integers

```
#define N 256  // elements per node
#define T (N * NUM_NODES()) // total elements
emu_chunked_array *A =
  emu_chunked_array_replicated_new(T, sizeof(long));
for (long j = 0; j < NUM_NODES(); j++) {
  long *P = emu_chunked_array_index(A, j * N);
  for (long i = 0; i < N; i++) P[i] = i;
```

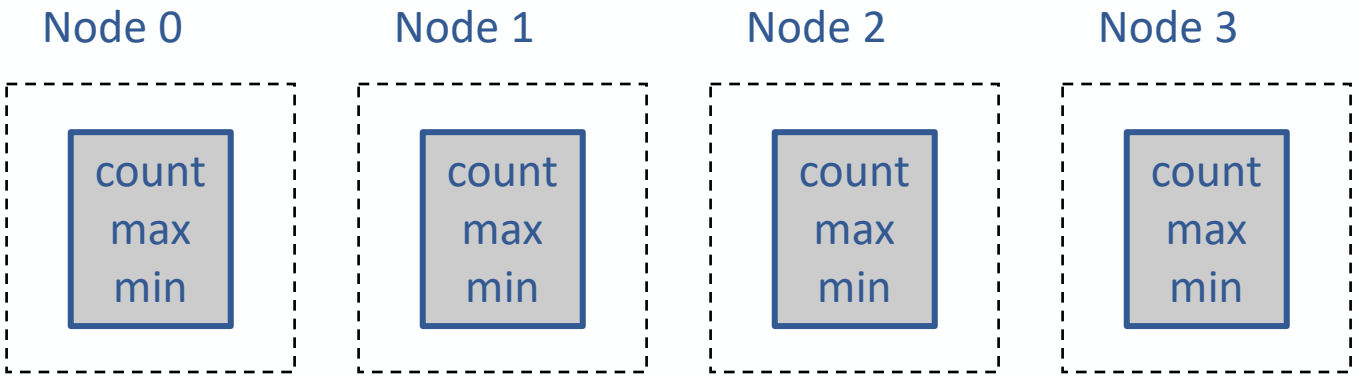
Chunked array allocated and initialized



Update Chunked Array: Worker Function

```
long *P = emu_chunked_array_index(A, begin);
for (long i = 0; i < end - begin; i++) {
  long score = scoreval(P[i]);
  if (score != 0) {
    ATOMIC_ADDM(&(s.count), 1);
    ATOMIC_MAXM(&(s.max), score);
    ATOMIC_MINM(&(s.min), score);
  }
}
```

Index into chunked array at thread starting point



Example: Intrinsics code with C utilities

```
#include <emu_c_utils/emu_c_utils.h>
…
int main(int argc, char **argv)
{
  emu_chunked_array *A = distr_array(); // chunked array
  initialize(); // initialize replicated data
  hooks_region_begin("example") // start profiling region
  // update replicated data
  emu_chunked_array_apply(A, GLOBAL_GRAIN_MIN(T, 64), update;
  // reduce replicated data
  long count = 0;
  long max = 0;
  long min = LONG_MAX;
  for (long i = 0; i < NUM_NODES(); i++) {
    struct stats *si = mw_get_nth(&s, i);
    count += si->count;
    if (si->max > max) max = si->max;
    if (si->min < min) min = si->min;
  }
  double time_ms = hooks_region_end();
  printf("time (ms) = %lf\n", time_ms);
}
```

Extra include file

Use hooks functions to mark start and end of region

Sample Program Execution: intrs_hook.c

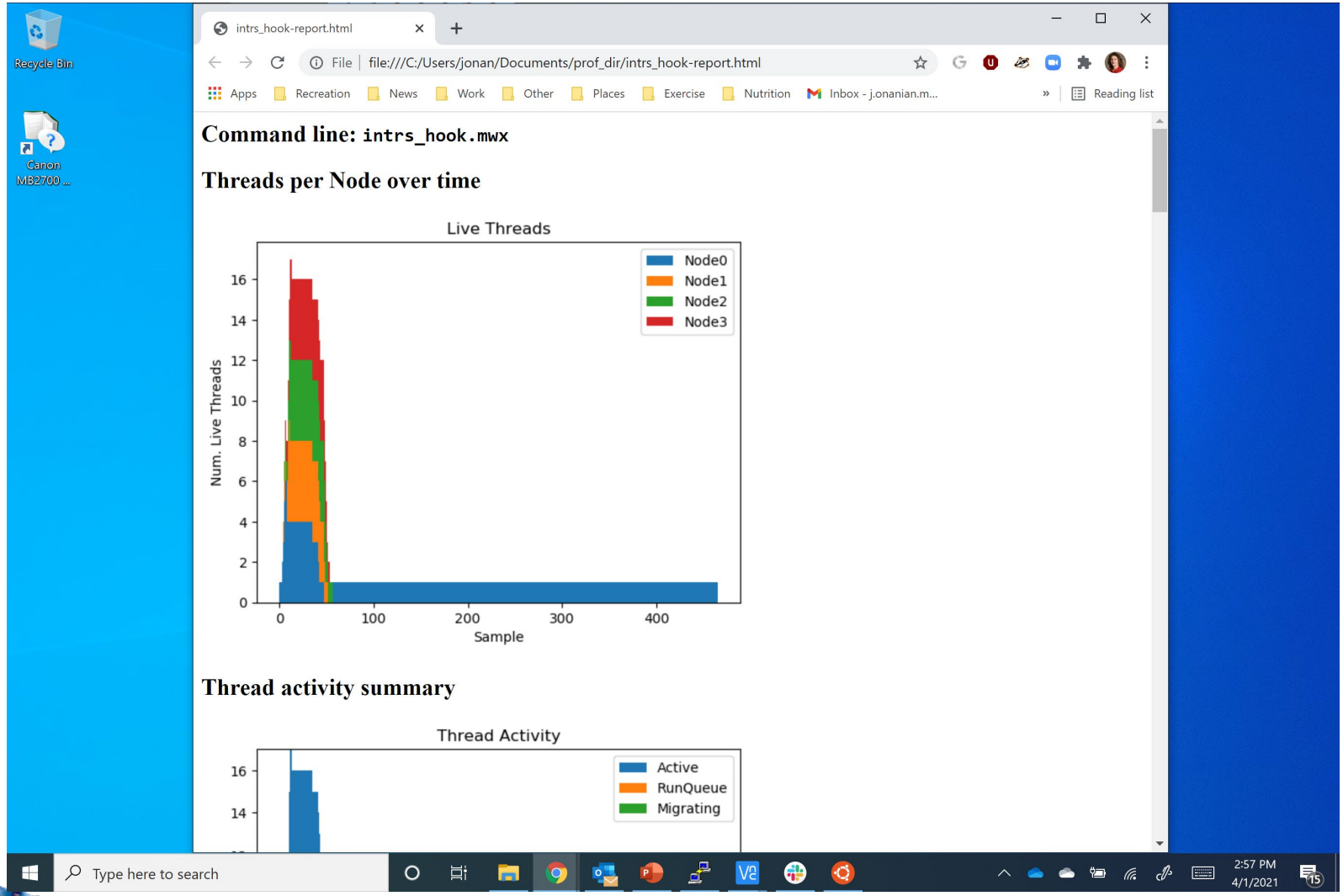
```
>>>>>>>> /usr/local/emu/bin/emu-cc  intrs_hook.c -o intrs_hook.mwx -l emu_c_utils
>>>>>>>> /usr/local/emu/bin/emusim_profile dir1 --total_nodes 4 -- intrs_hook.mwx
Generating profile in dir1/intrs_hook
emusim.x  --total_nodes 4
intrs_hook.mwx

        SystemC 2.3.3-Accellera --- Mar 24 2021 16:05:40
        Copyright (c) 1996-2018 by all Contributors,
        ALL RIGHTS RESERVED
Start untimed simulation with local date and time= Thu Apr  1 14:48:23 2021

End untimed simulation with local date and time= Thu Apr  1 14:48:23 2021

SysC Enumeration done. Program launching...
Simulation @0 s with local date and time= Thu Apr  1 14:48:23 2021

Simulation @1 ms with local date and time= Thu Apr  1 14:48:32 2021

Simulation @2 ms with local date and time= Thu Apr  1 14:48:41 2021

Simulation @3 ms with local date and time= Thu Apr  1 14:48:50 2021

{"region_name":"example","time_ms":0.60,"ticks":105481}
Simulation @4 ms with local date and time= Thu Apr  1 14:48:59 2021

time (ms) = 0.602749

Info: /OSCI/SystemC: Simulation stopped by user.
Generating dir1/intrs_hook_total_instructions.png
Generating dir1/intrs_hook_total_migrations.png
Generating dir1/intrs_hook.Thread_Enqueue_Map.png
Generating dir1/intrs_hook.Memory_Read_Map.png
Generating dir1/intrs_hook.Memory_Write_Map.png
Generating dir1/intrs_hook.Atomic_Transaction_Map.png
Generating dir1/intrs_hook.Remote_Transaction_Map.png
Generating dir1/intrs_hook.Live_Threads.png
Generating dir1/intrs_hook.Thread_Activity.png
Generating dir1/intrs_hook.MSP_Activity.png
Generating dir1/intrs_hook.SRIO_Outgoing_Activity.png
Generating dir1/intrs_hook.SRIO_Incoming_Activity.png
Report written to dir1/intrs_hook-report.html, you may open it in your browser now
```

Extra library linked

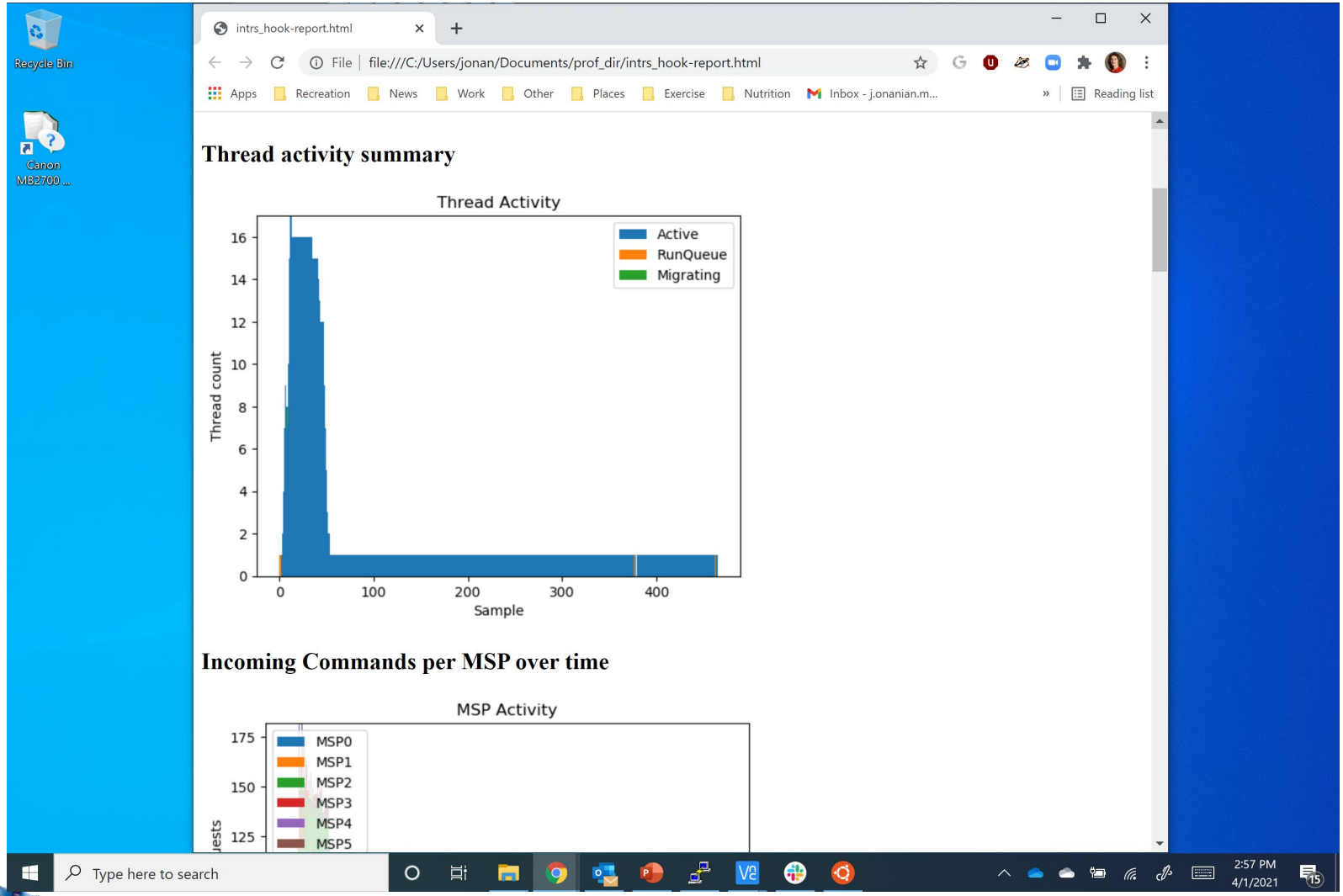
- Uses emusim_profile
- Simulator is run in timed mode
- Untimed portion is before the region start
- JSON output is at region end
- Profile images generated after simulation

Open html file in browser to see images

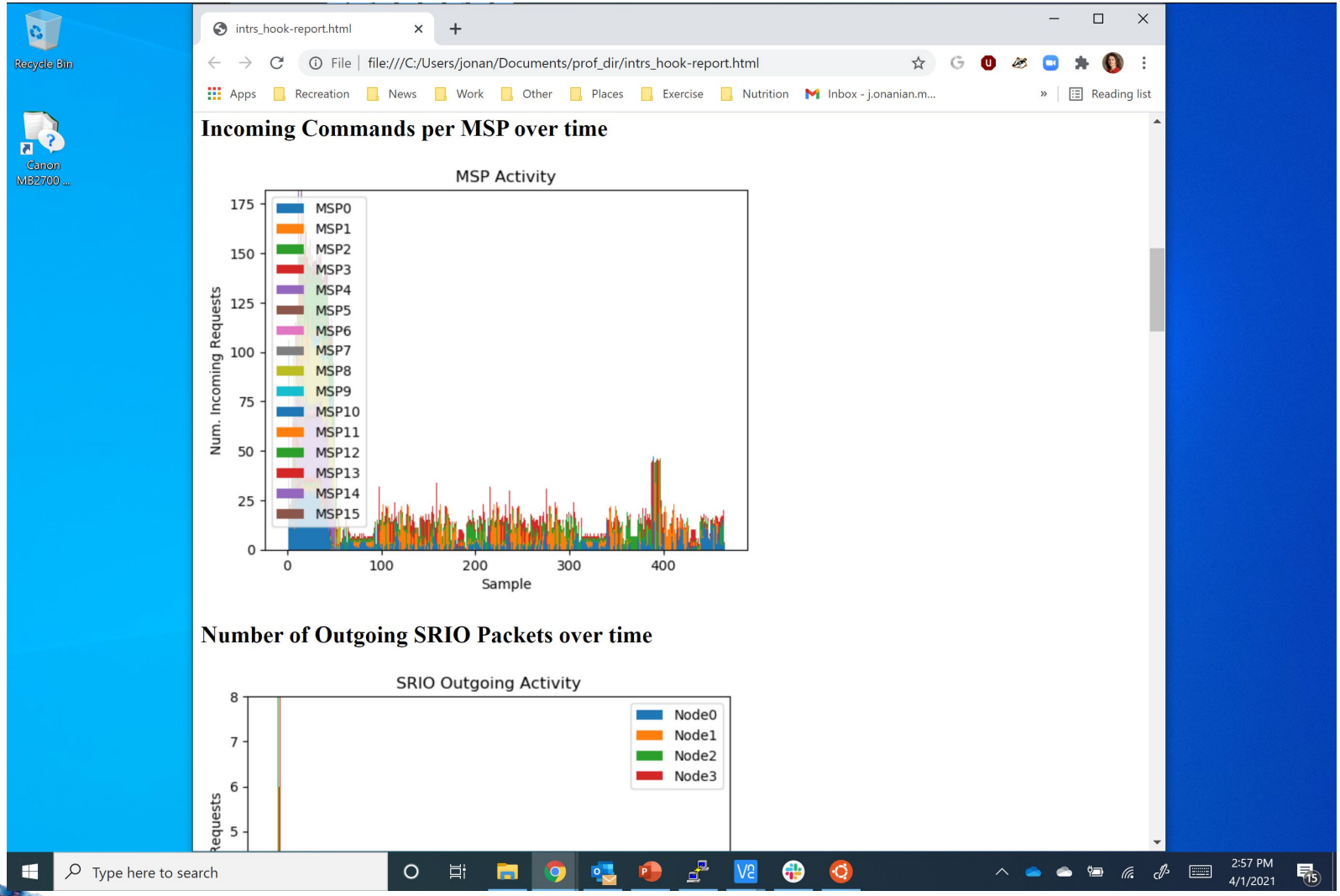
Profiler Output: Threads per Node



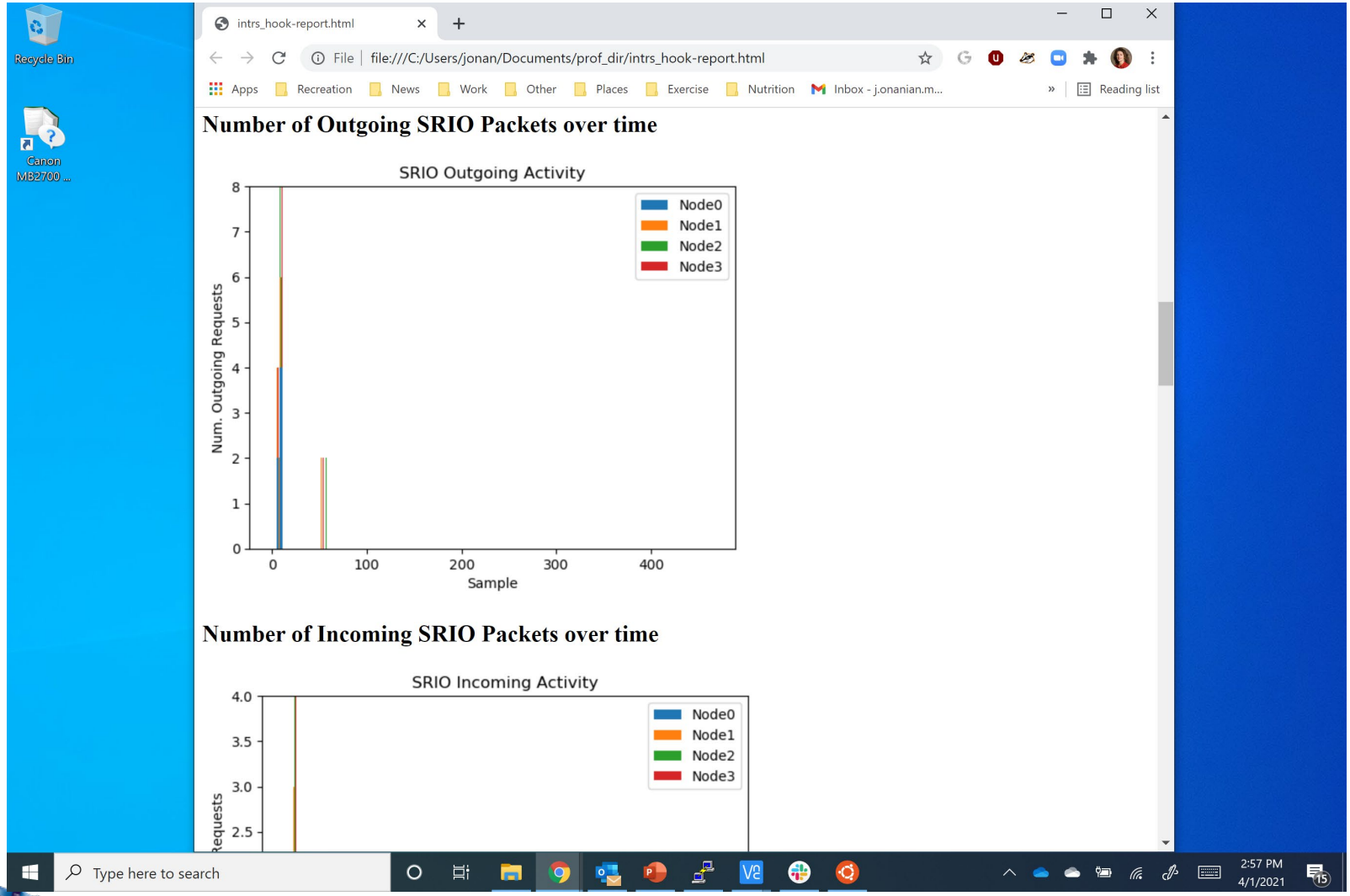
Profile Output: Thread Activity



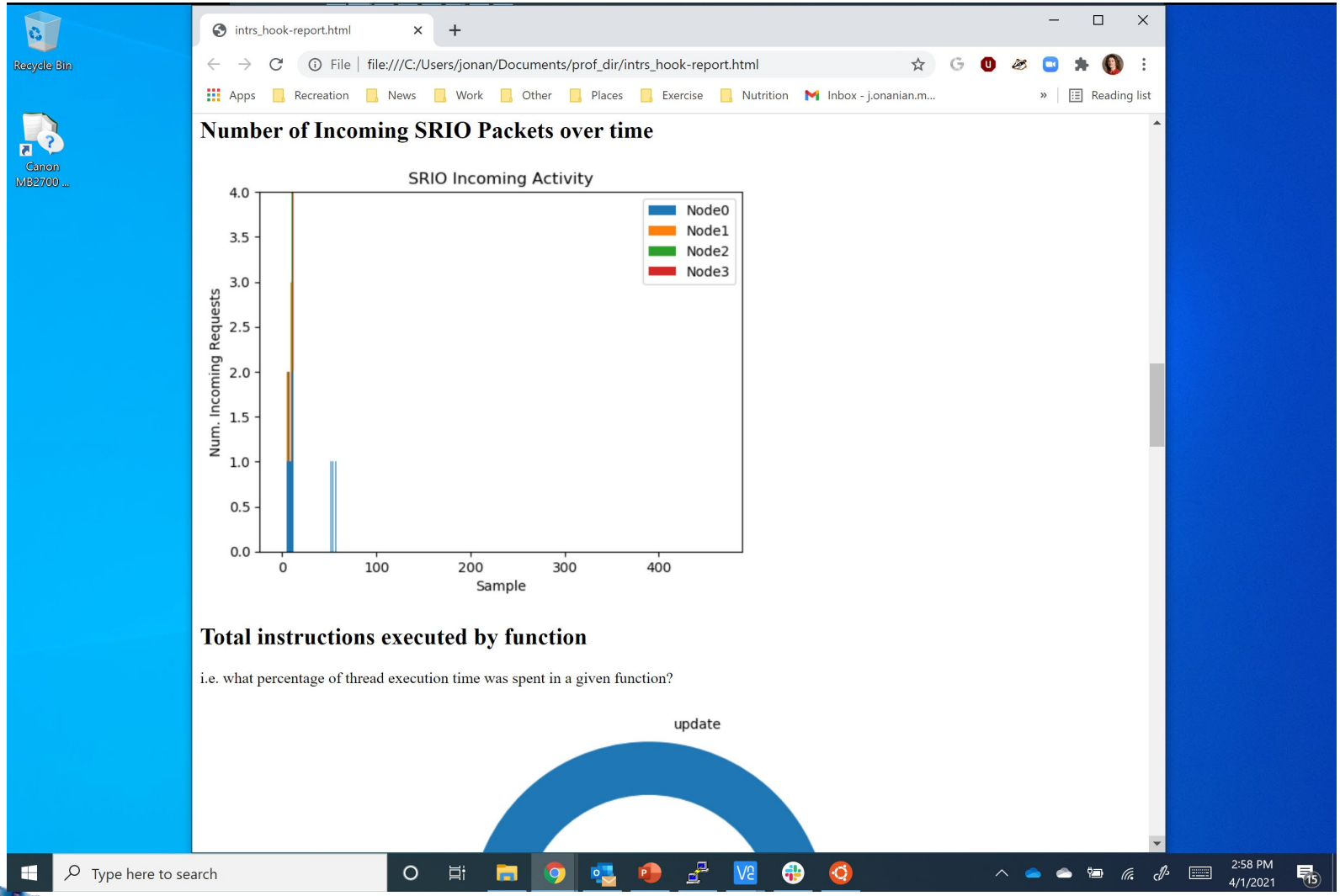
Profile Output: Incoming MSP Commands



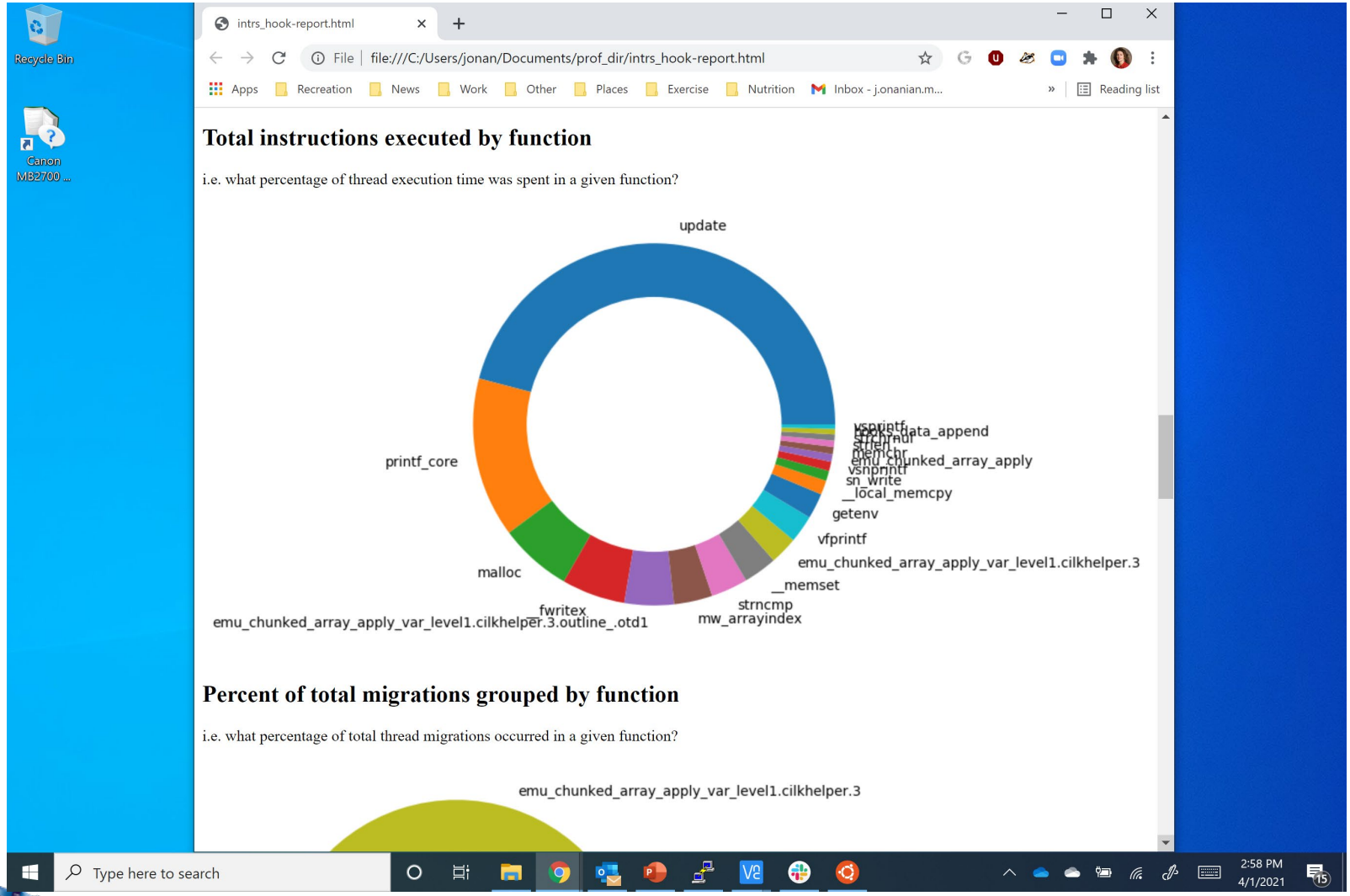
Profile Output: Outgoing SRIO Packets



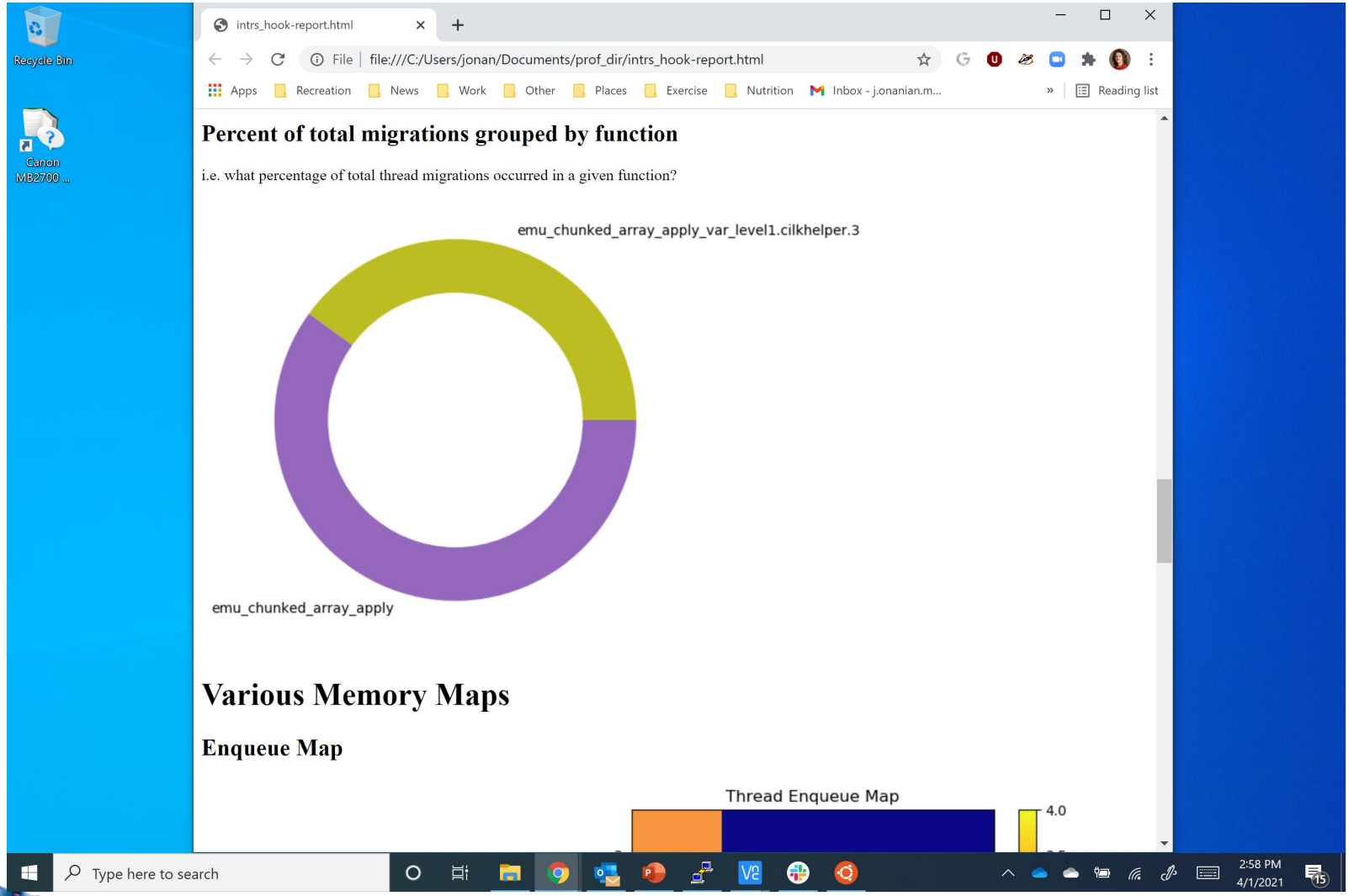
Profile Output: Incoming SRIO Packets



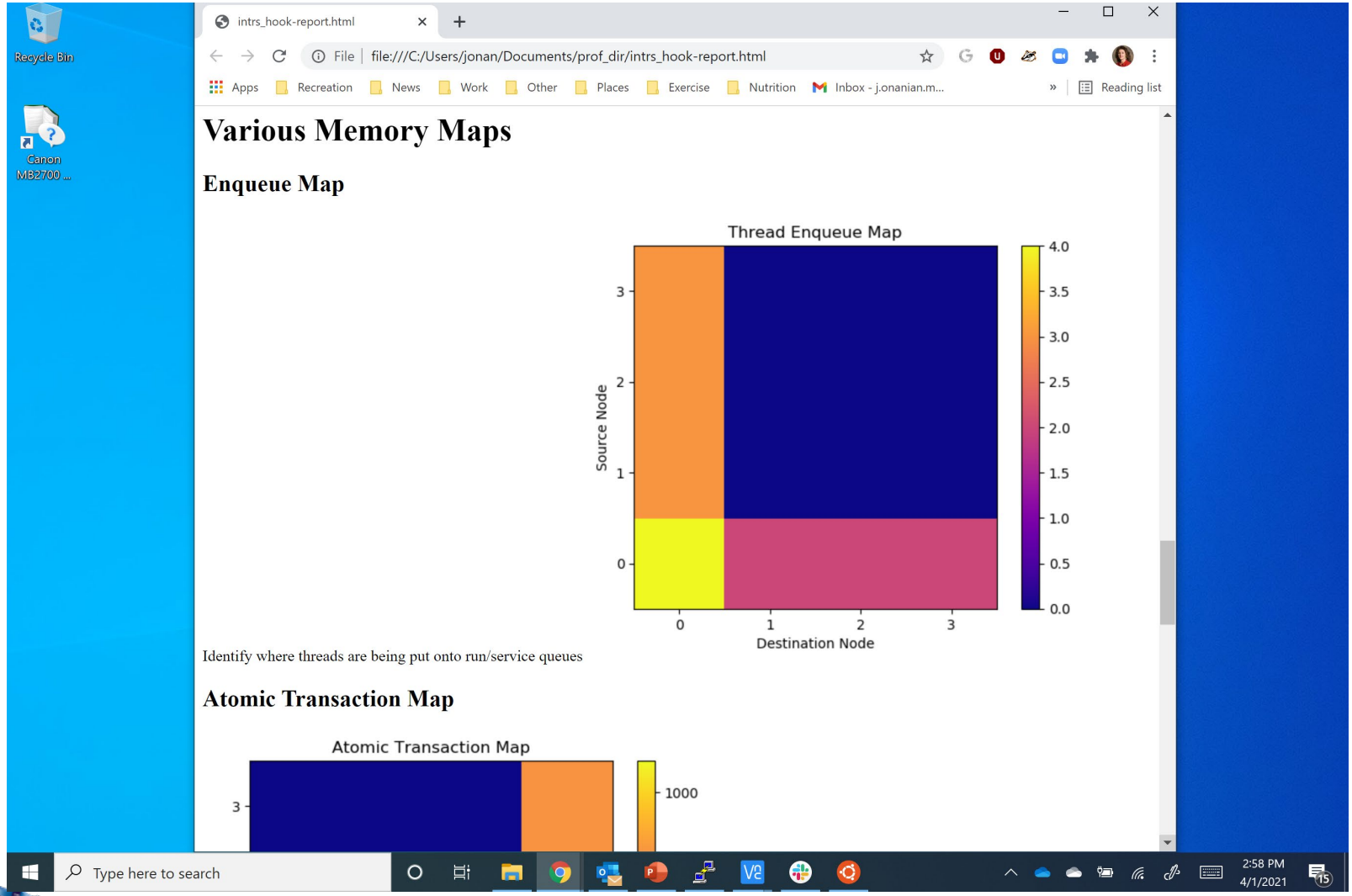
Profile Output: Instructions by Function



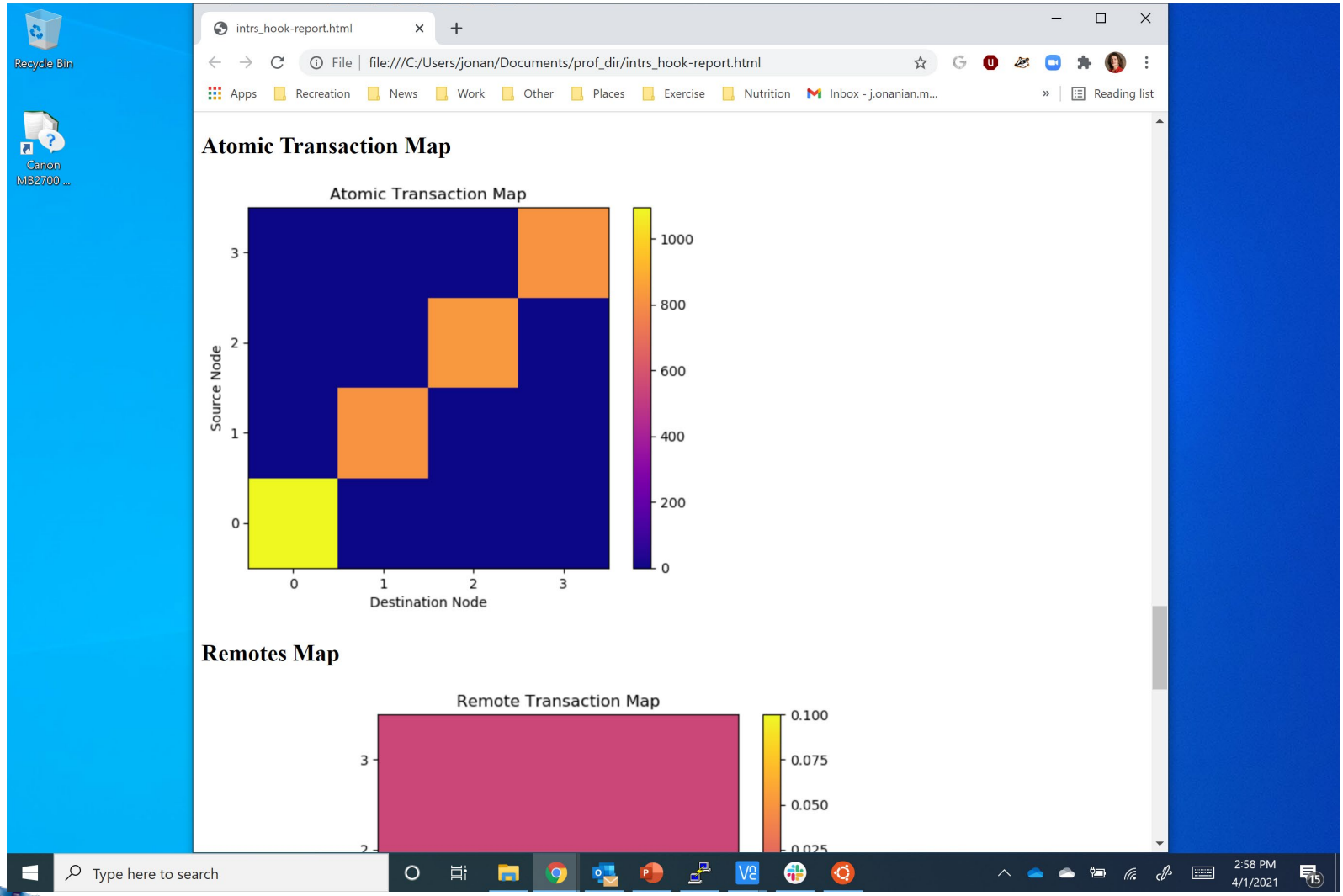
Profile Output: Migrations by Function



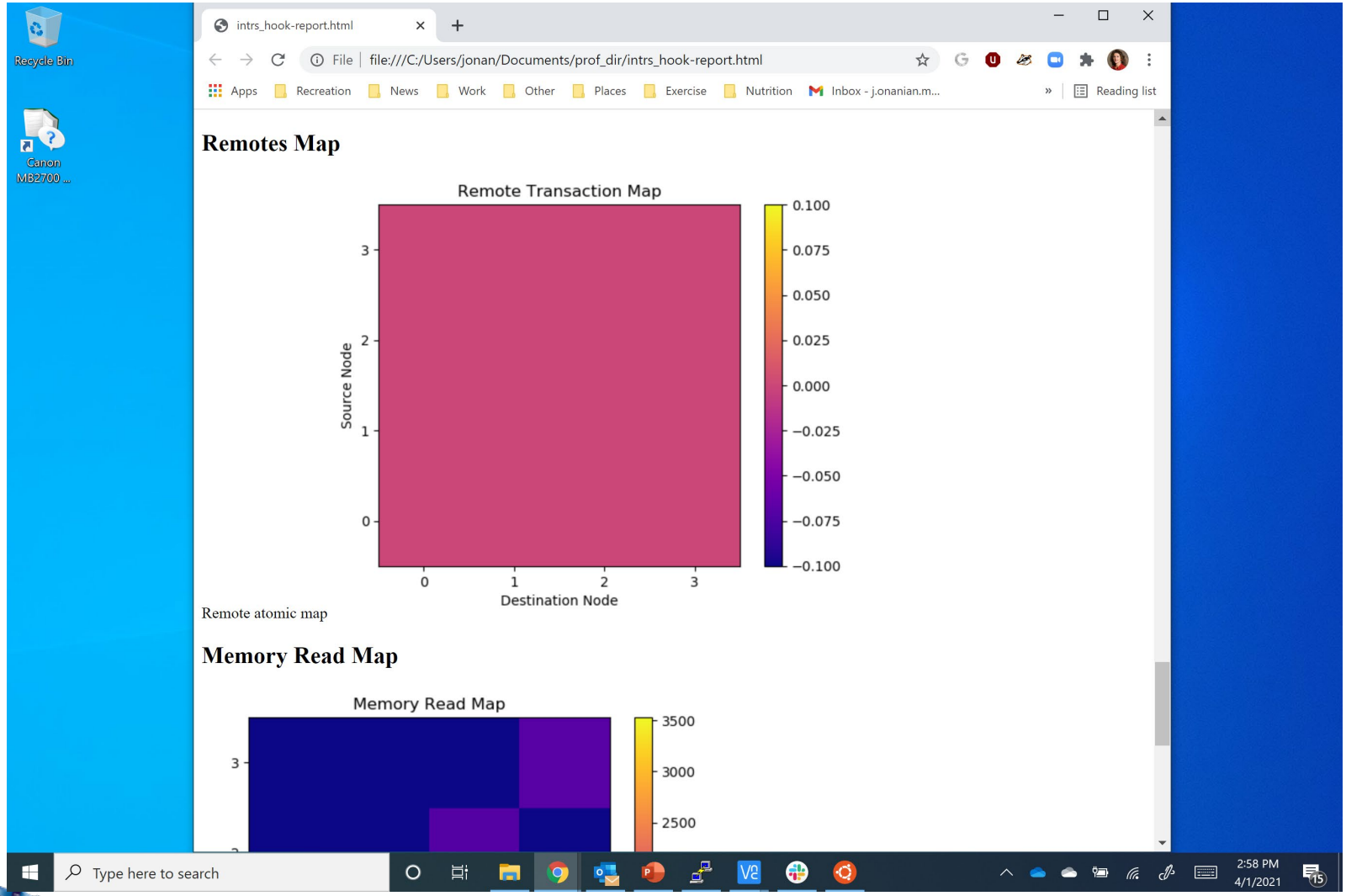
Profile Output: Enqueue Map



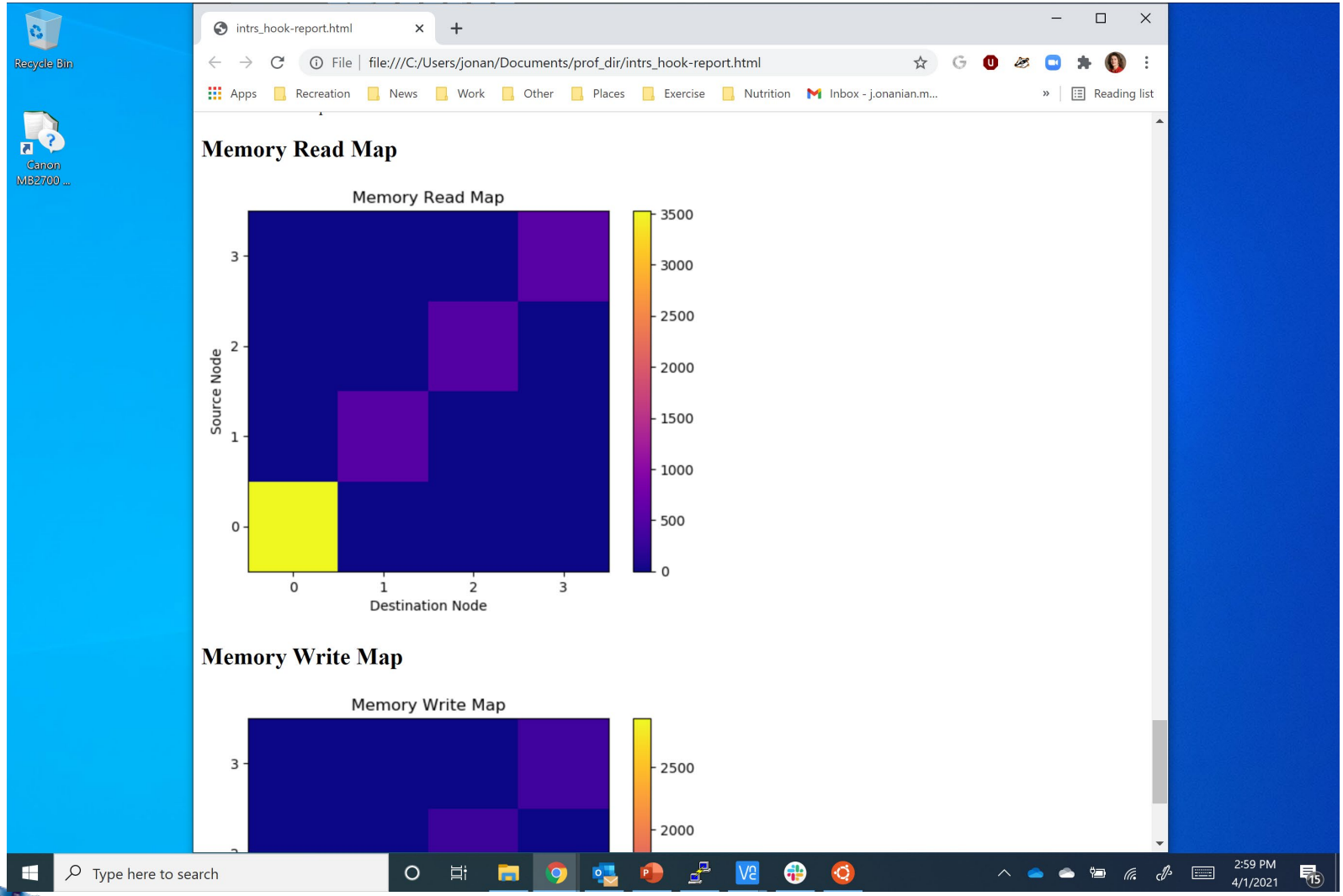
Profile Output: Atomic Transaction Map



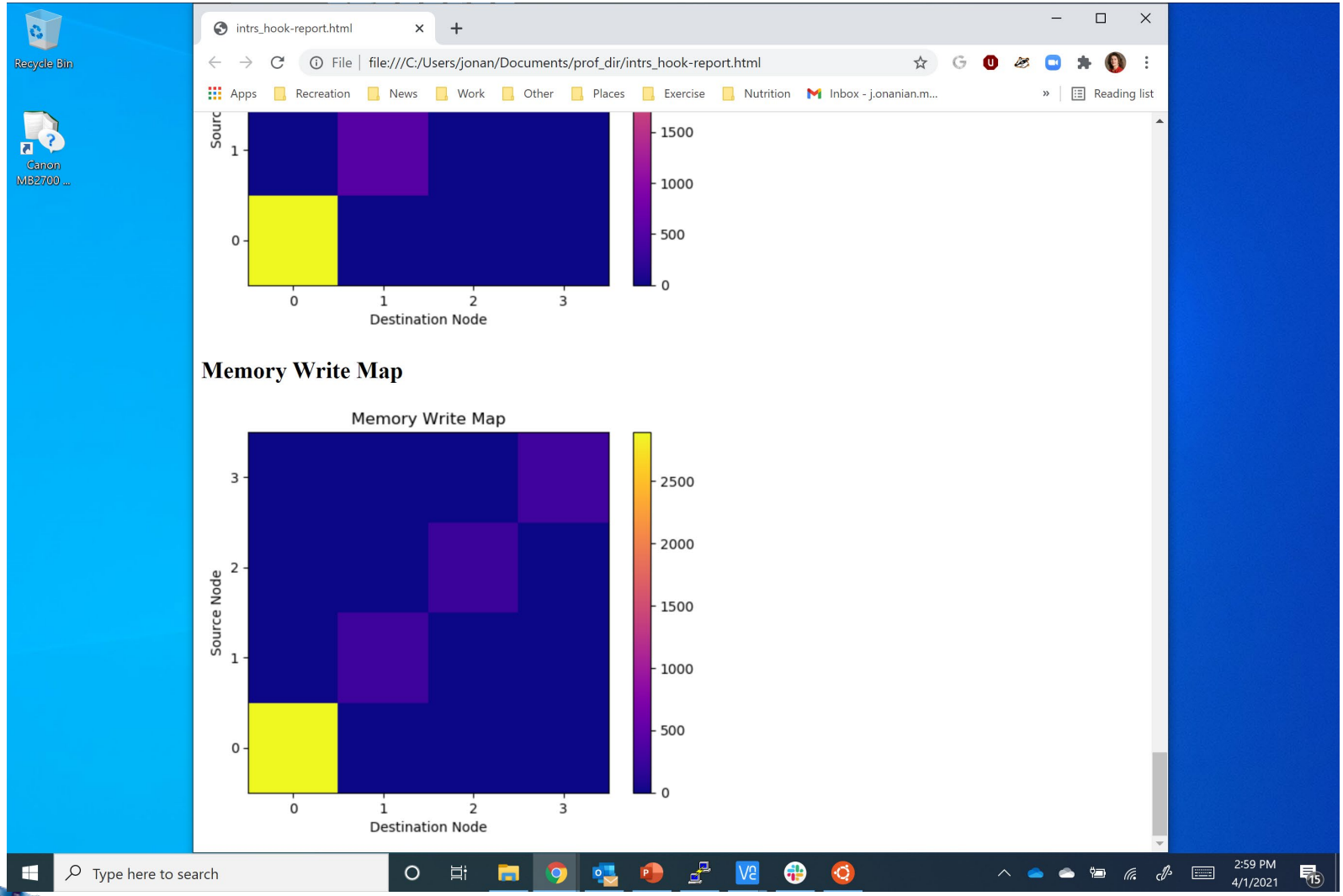
Profile Output: Remotes Map



Profile Output: Memory Read Map



Profile Output: Memory Write Map



Unit Summary: Measuring Performance

- Timing hooks and chunked arrays
- Profiling code
- Performance study

Exercises:
Re-write array initialization using chunked apply
Try striped and local arrays, study migration counts
Understand and verify profile images – are thread counts over time correct? How can we improve them?
Try using multiple regions and extra attributes to improve performance study