Das erste Schaubild zeigt Bandbreiten für die Migration von Datenmengen zwischen 4MB und 8000MB ohne Compileroptimierung. Der Container startet einen Thread auf jeder NUMA-Node, welche Ziel der Migration ist. Bei 8 Threads, werden die Daten also auf 8 NUMA-Nodes verteilt. Bei den Tests wurden auf einem System mit 8 NUMA-Nodes jeweils von NUMA-Node 0 aus die Daten verteilt. Das Schaubild zeigt, dass mit wenigen Threads die Migration der Seiten über hwloc deutlich schneller ist. Wenn mehr Threads verwendet werden, ist jedoch die Verion mit "first touch" schneller.

Das liegt vermutlich daran, dass bei steigender Threadanzahl die Arbeit pro Thread kleiner wird, währen die syscalls von hwloc alle im Bottleneck des Kernels "stecken bleiben". Schaubild 2 zeigt, ab welcher Zahl Threads bei eine größe von 8GB die Migration mit "first touch" schneller ist.

Schaubild 3 und 4 zeigen den selben Versuchsaufbau, allerdings mit Compileroptimierung, `g++ -O3`. Dabei zeigt sich deutlich, dass die Variante mit hwloc nicht von der Optimierung profitiert, bei der Variante mit "frist touch" aber bereits bei einem Thread eine deutlich höhere Bandbreite erreicht wird.

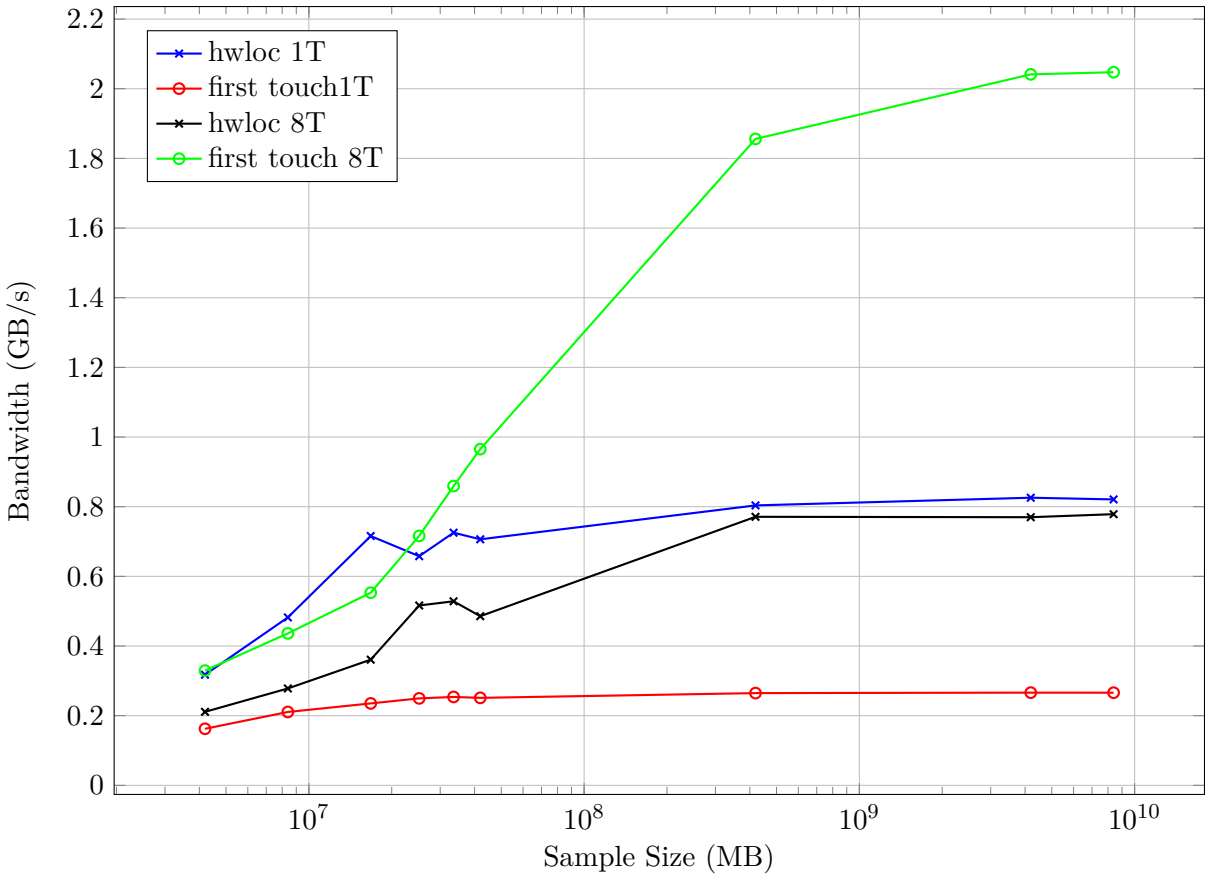

Figure 1: Migration performance, depending on problem size

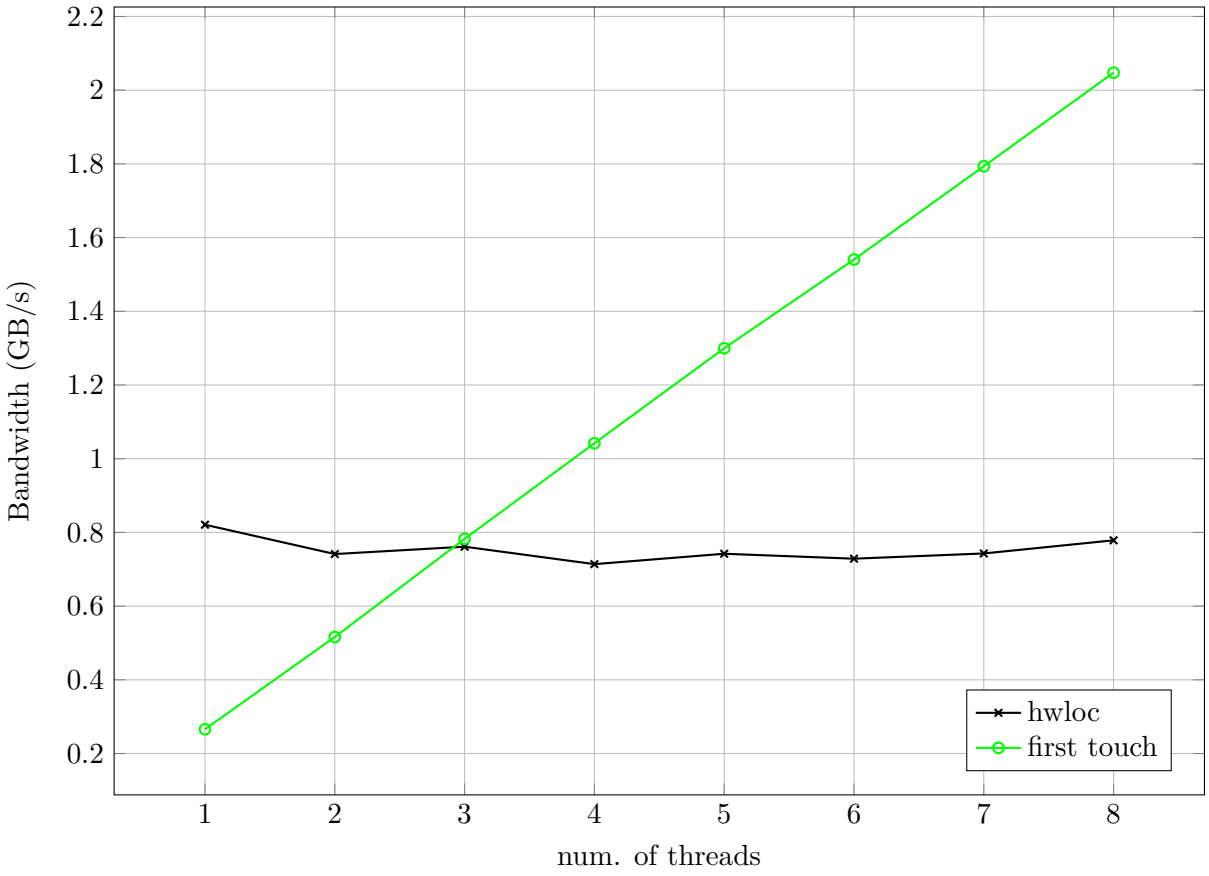

Figure 2: Migration performance, depending on number thread. Sample Size is 8GB. Num. of threads also represents number of distribution domains. (4 threads mean, data is also distrubuted to 4 domains)

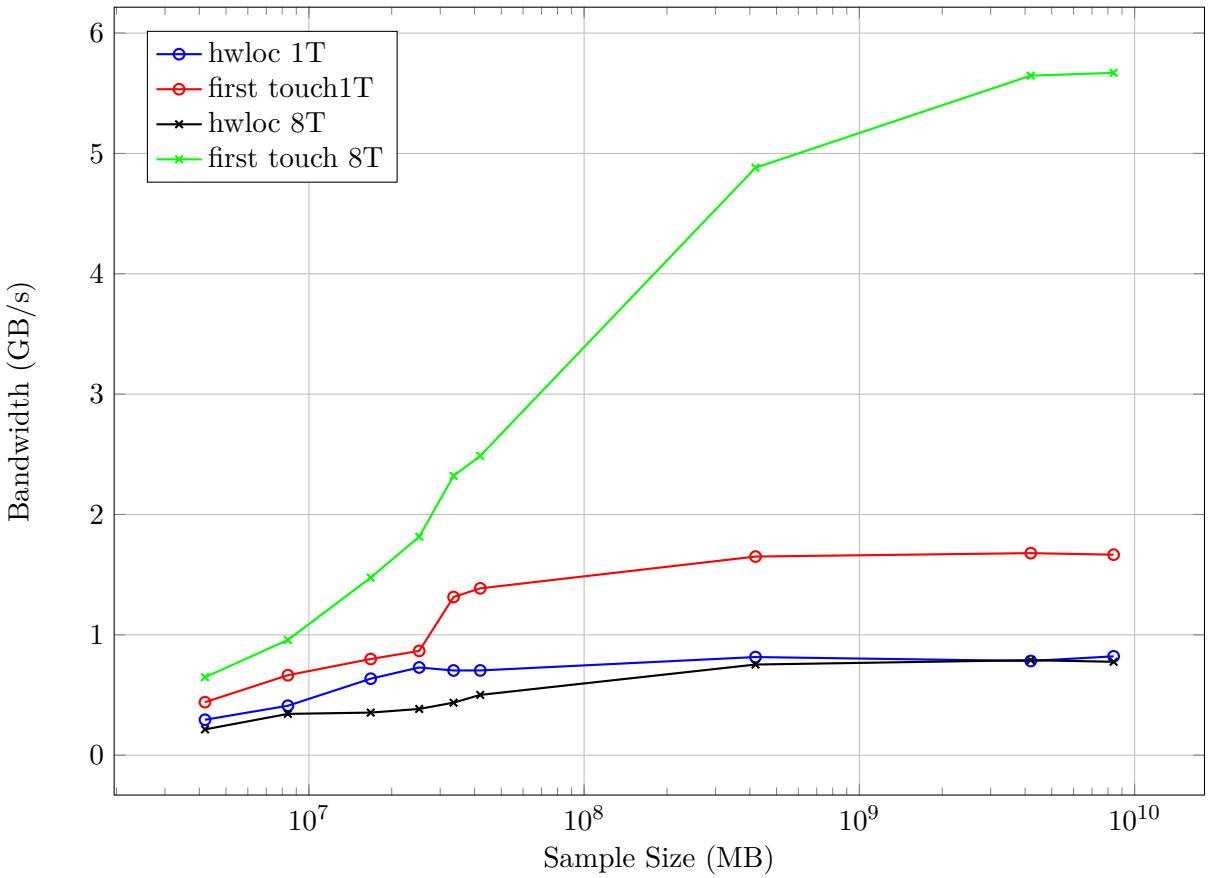

Figure 3: Optimized migration performance, depending on problem size

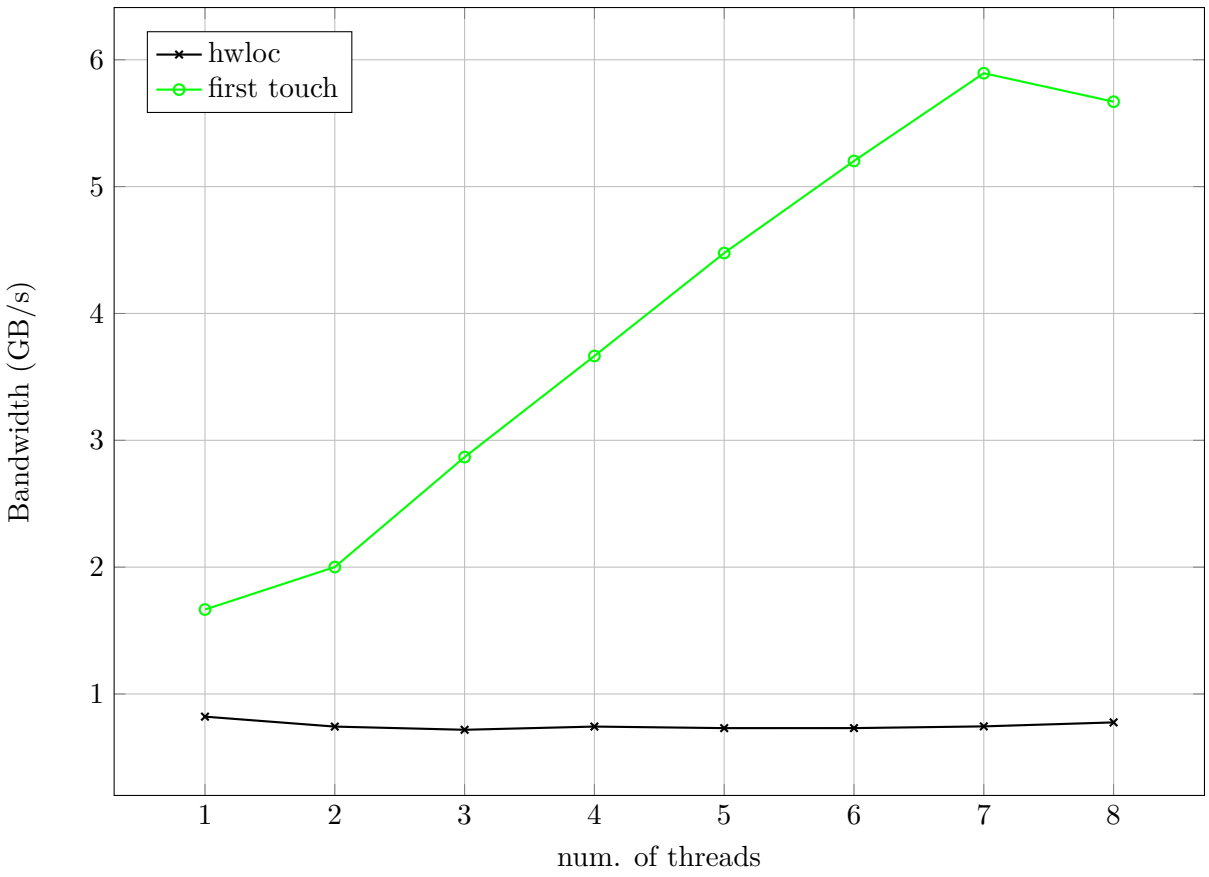

Figure 4: Optimized migration performance, depending on number thread. Sample Size is 8GB. Num. of threads also represents number of distribution domains. (4 threads mean, data is also distrubuted to 4 domains)

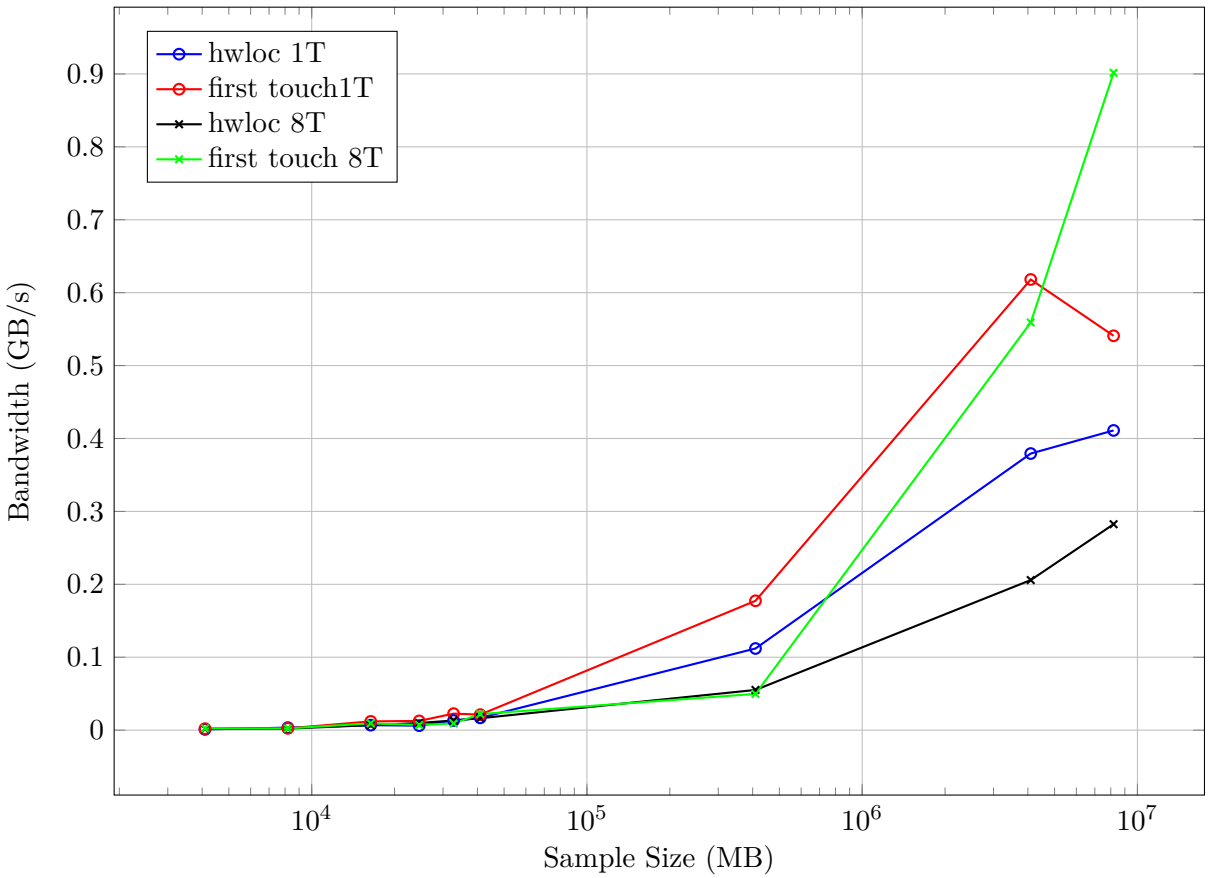

Figure 5: Migration Bandwidth, kB Range