BMC Bioinformatics

RESEARCH

Open Access

Optimize data-driven multi-agent simulation for COVID-19 transmission

Chao Jin[1*], Hao Zhang[2,3], Ling Yin[2], Yong Zhang[2] and Sheng-zhong Feng[1]

*Correspondence:
jinchaohpc@gmail.com

[1] National Supercomputing Center in Shenzhen, Shenzhen 518055, Guangdong, People's Republic of China
[2] Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, Guangdong, People's Republic of China
[3] University of Chinese Academy of Sciences, Beijing 100049, People's Republic of China

Abstract

Background: Multi-Agent Simulation is an essential technique for exploring complex systems. In research of contagious diseases, it is widely exploited to analyze their spread mechanisms, especially for preventing COVID-19. Nowadays, transmission dynamics and interventions of COVID-19 have been elaborately established by this method, but its computation performance is seldomly concerned. As it usually suffers from inadequate CPU utilization and poor data locality, optimizing the performance is challenging and important for real-time analyzing its spreading.

Results: This paper explores approaches to optimize multi-agent simulation for COVID-19 disease. The focus of this work is on the algorithm and data structure designs for improving performance, as well as its parallelization strategies. We propose two successive methods to optimize the computation. We construct a case-focused iteration algorithm to improve data locality, and propose a fast data-mapping scheme called hierarchical hash table to accelerate hash operations. As a result, The case-focused method degrades ∼ 90% cache references and achieves ×4.3 speedup. Hierarchical hash table can further boost computation speed by 47%. And parallel implementation with 20 threads on CPU achieves ×80 speedup consequently.

Conclusions: In this work, we propose optimizations for multi-agent simulation of COVID-19 transmission from aspects of algorithm and data structure. Benefit from improvement of locality and multi-thread implementation, our methods can significantly accelerate the simulation computation. It is promising in supporting real-time prevention of COVID-19 and other infectious diseases in the future.

Keywords: Multi-agent simulation, Case-focused method, Hash table

Background

Multi-agent simulation (MAS) is an essential technique for exploring system phenomena in which the overall behaviour is determined by the constituent autonomous entities [1]. It provides an effective tool for modeling systems with complex organization and nonlinear interactions. As agents can be applied with different disciplines, it is widely used in studying social, economic, organization and epidemiology sciences [2–4]. Domínguez et al. [5] propose multi-agent modeling for complex supply chains to overcome limitations from classical methods. McArthur et al. [6] investigate MAS technology in power

industry with complex scalable power networks. Moreover, it is also evaluated as efficient in smart city, such as controlling city congestion, pollution and delivery time [7]. Many other applications can be found in [8].

A significant application of MAS is studying infectious diseases. Classical methods use mathematical models to emulate the transmission of infectious diseases. Some classical examples include Susceptible-Infect-Susceptible (SIS) epidemic model [9], Susceptible-Infect-Recovered (SIR) epidemic model [10], etc. However, these simple models are not afforded to analyze complex and find-grained systems. MAS integrating evolution and phylogeny helps to understand emerging infectious diseases in complex systems. Dion et al. [11] leverage it to study the landscape epidemiology of the foot-and-mouth disease in South Africa. Yergen et al. [12] propose IDESS for rapidly constructing MAS models of Avian Flu (H5N1) virus spreading.

Outbreaks of COVID-19 raise concerns about effectively preventing spread of infectious diseases, and MAS is widely adopted for studying this issue. Its fine-grained spreading dynamics is established through many efforts. Castro et al. [13] analyze the spread processes of COVID-19 epidemics in open regions by considering effects from different environments. Vyklyuk et al. [14] propose modeling its spread in large regions by simulating a set of autonomous multi-agent systems. Nanna et al. [15] extend MAS to dynamically verify influences on diseases spread from government strategies. For COVID-19 preventions, both non-pharmaceutical interventions (NPI) and pharmaceutical interventions (PI) have been elaborately invested [16] with this paradigm. Yin et al. [17] proposed a data-driven NPI MAS model to suppress the diseases in Shenzhen and evaluate strategies including contact tracing, mask wearing and prompt testing. Zhou et al. [18] examine the spatial heterogeneity of the disease transmission and optimize vaccine distribution strategies considering spatial prioritization.

Although many researches focus on exploring underlying dynamics through MAS modeling, fewer concern the computation performance [19–21]. However, computation performance is a significant aspect for large-scale simulations where obstructions from heavy calculation, intensive memory access and communication are inevitable. In general, approaches to improve MAS performance include parallel computing [22–24] and distributed computing [25–27] on both CPU and GPU platforms. Well known of parallel and distributed platforms contain Mason [28], Gama [29] and Simphony [30]. Meanwhile, FPGA is remarked with fine-grained parallelism and flexible memory architecture. Some studies also focus on the acceleration on FPGA platforms [31, 32]. As communication patterns among agents are continuously changing in distributed systems, some works seek for effective agent allocation strategies to improve the performance [33, 34]. On the other hand, massive-scale simulations in serial algorithm often suffer from poor data locality, but seldom researches about this issue is established. Willem et al. [35] introduce a sorting phase of population and optimize data structure to improve system performance.

A general MAS simulation [36] for infectious diseases contains two steps: (a) update each person's health state according to the epidemic model; (b) compute the disease transmissions over the contact network. Accessing agent states is irregular and random, which is the major factor limiting the computational efficiency. In this work, we study optimizing MAS for COVID-19 transmission in Shenzhen. Two methods are explored

and evaluated including algorithm and data structure optimization. In order to improve data locality, we reconstruct the loop order of MAS algorithm, and propose a hierarchical Hash table structure leveraging cache and hardware properties. Our results show prominent improvement in the system performance and indicate wide applicability to interventions for infectious diseases.

Specifically, we make the following contributions:

(1) We reorder the inner structure of MAS method to improve the data locality, where iterations of infectious agents take priority rather than the time evolution. And formulate a simple convolutional scheme to eliminate its systematic errors. (Section Methods–Case-focused simulation)
(2) We propose a hierarchical hash table to support irregular and randomly accessing agent states with high efficiency. It leverages cache characteristics to organize data in a compact manner and reduce its sparsity in memory space, and adopts a key-value separating structure for flexible operations. In addition, single instruction, multiple data (SIMD) instructions are applied to boost the handling speed of hash collisions. (Section Methods– Hierarchical Hash Table)

Methods

Epidemic dynamics

A stochastic, discrete-time susceptible-latent-infectious-removed (SLIR) model is implemented where the transmission of COVID-19 is triggered by contacts between agents in households, workplaces, schools and other buildings [17]. Once a susceptible individual has a contact with an infectious agent, the probability of infection p via this contact is calculated as follows:

$$p = pTrans \times I_c \times r \tag{1}$$

where *pTrans* denotes the transmission probability per contact and is estimated as 0.165 by calibrating the modeled basic reproductive number R_0 to the observed value of 2.4 [37–41]. I_c is the intensity of daily contact at different contact settings derived from a contact survey [42]. r differentiates the infectivity of infectious agents with and without symptoms, i.e. the infectivity of asymptomatic agents is as 0.12 of their counterparts [17]. In addition, this simulation assumes that all infected agents would not be re-infected.

The infectious dynamics of SLIR model is demonstrated in Fig. 1. Once a susceptible individual (S) is infected, a probability of 25% is assumed to turn into latent status [43–45]. The latent period (La) is set to 4.6 days (λ) for asymptomatic agents to become infectious [46–48]. These agents remain infectious for 9.5 days (μ) until being removed

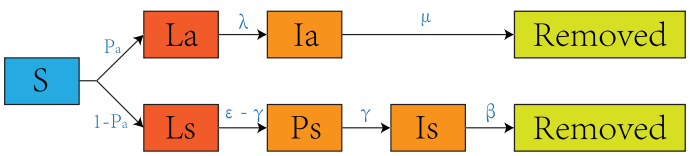

Fig. 1 A compartmental SLIR model for COVID-19

from the model after recovery [49]. Symptomatic agents are assigned an incubation period (ϵ) with a mean of 5.2 days to manifest symptoms (Is), including latent status (Ls) [48]. Their infectivity starts from 2 days (γ) before symptom onset (Ps) [47]. After the onset of symptoms, agents remain infective until they get recovered. Parameters in this SLIR model are confident based on the assessment of local Center for Disease Control (CDC) collaborators, who have first-hand COVID-19 clinical data.

Case-focused simulation

In general, MAS algorithm for infectious diseases makes efforts to mimic its natural transmission process among population, where diseases spread as time evolves. Hence, traditional MAS algorithm is evolution-focused, where the evolution is represented as calculation by time step, and is shown in Algorithm 1. It contains 3-order loops. The first loop ($k \in MaxIter$) is the time-evolving loop, where *MaxIter* is simulated max time step. The second loop is made up by three subprocesses (two of $a_i(k) \in A(k)$, and one of $b_j(k) \in B(k)$), including removing recovered agents, building up contacting networks, implementing diseases spreading calculation. A, B organized as hash table format denote infected agents and contacting network respectively. *Regions* represents general venues population gather together including both private and public areas. The innerest loops ($g_i \in Regions(a_i)$ and $l_j \in b_j(k)$) scan over infected agents' appearance places daily to find contacting population and contacted candidates to decide increasingly infected agents at current time step k. These agent information are dynamically changed with either shrinking or expanding behaviors in hash tables A and B during the iteration.

Algorithm 1 Evolution-focused MAS for COVID-19 transmission

Input: $A, B, MaxIter, Regions.$
Initialize: $A(0) \leftarrow InitializeSeedAgents()$

while $k < MaxIter$ **do**
  for $a_i(k) \in A(k)$ **do**
    $r_i \leftarrow GetRecoverDate(a_i)$
    if $k > r_i$ **then**
      $A(k) \leftarrow DeleteFromA(a_i)$
  for $a_i(k) \in A(k)$ **do**
    for $g_i \in Regions(a_i)$ **do**
      $n_i \leftarrow GetNumSampled(g_i, a_i)$
      $l_i \leftarrow SampleContactAgents(g_i, n_i)$
      $b_i(k) \leftarrow [b_i(k), l_i]$
    $B(k) \leftarrow InsertToB(b_i(k))$
  for $b_j(k) \in B(k)$ **do**
    for $l_j \in b_j(k)$ **do**
      $Prob_j \leftarrow GetInfectedProbability(l_j)$
      if $rand < Prob_j$ **then**
        $a_j(k+1) \leftarrow GetDisease(l_j)$
        $A(k+1) \leftarrow InsertToA(a_j(k+1))$
  k+=1;

Algorithm 2 Case-focused MAS for COVID-19 transmission

Input: $A, Regions.$
Initialize: $A \leftarrow InitializeSeedAgents();$
$MaxCase \leftarrow GetSize(A);$

while $m < MaxCase$ **do**
  $a_m \leftarrow A(m)$
  $[T_{m0}, T_{m1}] \leftarrow GetInfectiousDate(a_m)$
  for $g_i \in Regions(a_m)$ **do**
    for $t_m \in [T_{m0}, T_{m1}]$ **do**
      $n_m \leftarrow GetNumSampled(g_m, a_m)$
      $l_m \leftarrow SampleContactAgents(g_m, n_m)$
      $b_m \leftarrow [b_m, l_m]$
  for $l \in b_m$ **do**
    $Prob_l \leftarrow GetInfectedProbability(l)$
    if $rand < Prob_l$ **then**
      $a_l \leftarrow GetDisease(l)$
      $A \leftarrow InsertToA(a_l)$
  $A \leftarrow DeleteFromA(a_m)$
  $MaxCase \leftarrow GetSize(A)$

As indicated by middle-order loops in this algorithm, all the elements from the constructed hash tables A and B require accessing and utilization once for each time iteration. As the amount of population is large, such a scanning mechanism wastes data fetching from memories. It will degrade system performance severely owing to this poor locality as data arrangement heavily exceeds cache capacities. For simulation in a megacity with ones of millions population, this issue is inevitable in performing real-time simulation and quickly response to intervention strategies. Key point is to manage effective reutilization of data as fetching from hash tables to maintain the locality.

In order to improve data locality, we propose a case-focused method, where the algorithm loops are reorganized. As shown in Algorithm 2, the table scanning loop is elevated to the outer order, while time-evolution loop is demoted. *MaxCase* is size of A, and changes automatically as A varies during simulation. At a primitive transmission stage of pandemic diseases, rapid spreading among population increases *MaxCase* prominently. It will get decreased and vanish when diseases are suppressed and immunity gets common. In the reconstructed loops, infected agent a_m gets accessed and transferred to cache once for each iteration. The time evolution is implemented in the inner loop $[T_{m0}, T_{m1}]$, thus a_m can be reutilized $T_{m1} - T_{m0}$ times. Meanwhile, as infection period of a_m finishes at the end of each agent m, m gets recovered naturally and we need not to query for recovered agents appearing at evolution-focused method. In addition, contacting candidates b_m are sampled independently for each infected agent a_m, and construction of B is independent. As bulky information of agents will be transferred into higher-level cache and frequently fetching from main memory is suppressed, it is expected that case-focused method will improve system performance significantly.

It should be mentioned that these two algorithms have similar computation complexity possessing 3-order loops. An interesting difference between them is causality. The evolution-focused method follows a natural process, while case-focused method reduces priority of time evolution to maintain data locality. It will affect produced

distribution of infected-agent evolution. For widely spreading diseases with large overlapping susceptible population from infectious agents, the probability of a candidate b_i gets infected is boosted by both of the amount of its contacting infectious agents and their infectious periods. The calculation order of the two factors make different effects, where the inner loop of factor takes priorities. For each time step k in evolution-focused MAS, number of contacting agents take the priority to determine candidate's healthy status. However, in case-focused MAS, each candidate agent m is domained by infectious agent's period $[T_{m0}, T_{m1}]$. Both of them share a same infectious probability, but candidate b_m is more likely to be infected within a relatively delayed time with respect to evolution-focused method.

We formulate this difference as the systematic error from case-focused method. It can be demonstrated and modified by a simply assumption. Assuming the delayed infected time δ follows a uniform distribution, it can be directly derived from statistic theory that the average delayed time $\bar{\delta}$ obeys a gaussian distribution $N(\mu, \sigma^2)$. μ represents the mean of the distribution, and σ^2 is its variance, which can be regarded as super parameters tuned for real simulation. Conversion between two algorithms can be implemented by convoluting former method with a delayed-gaussian function. In other words, case-focused MAS will generate delayed and vague results with respect to traditional method.

Hierarchical hash table

As indicated in algorithms 1 and 2, hash table is an important participant in the simulation, and its performance affects system significantly. In general, hash tables consume the majority of cycles on many key applications such as databases [50], networking [51] and genomics [52]. But they suffer from inefficiencies in current systems owing to poor core utilization and poor spatial locality [53]. Hash tables spread key-value pairs uniformly and sparsely across allocated memory to reduce mapping conflicts. In the MAS simulation, each agent's index and its bulky information are formed together into the

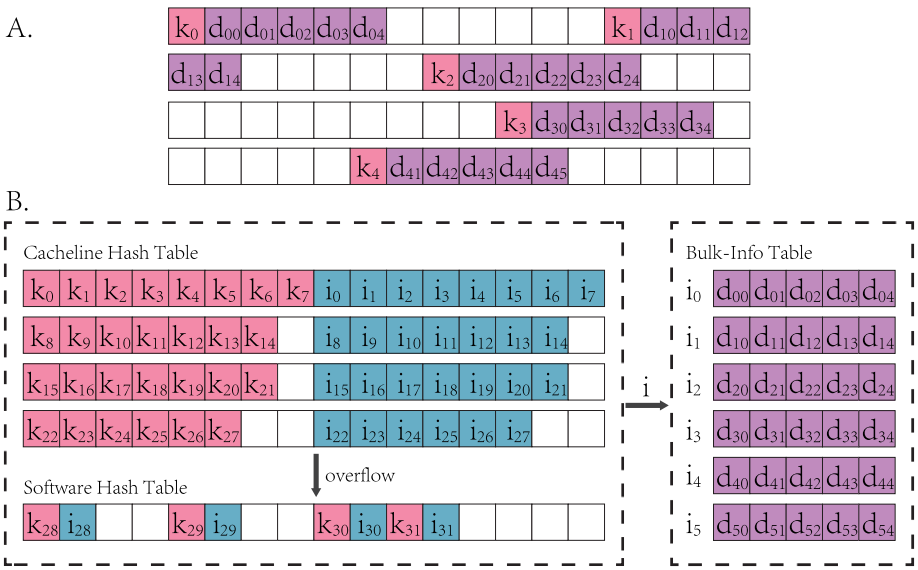

Fig. 2 **A**: Traditional (software) hash table format. **B**: Overview of hierachical hash table format

key-value pair filled in the table as shown in Fig. 2A. Due to sampling for agents is a random process during the simulation, same-line neighbors of frequently accessed agents may be rarely accessed. This leads to a significant waste of cache capacity.

There are many researches focusing on reducing hash table overheads. Data-level parallelism of inter keys is introduced in prior works to optimize the throughput [50, 54]. Near-memory [55] and near-storage [56, 57] acceleration bypass the cache hierarchy entirely to avoid spatial locality problems. Exploiting hierarchical memory layout and characteristics of caches [53] is found improving spatial locality prominently.

In the MAS computation, the key-value pair of agent information is made up by agent index (key, k) and bulky data (value, d) derived from simulation, such as infected and recover date. These pairs stored by traditional (software) hash tables are allocated sparsely across memory as demonstrated in Fig. 2A. Lookup and update operators of hash table have to access bulk memory in units $k + d$, which generate larger memory occupancy and heavier accessing burden. Hence, cross-line and misses of caches are frequently confronted. It is essential to allocate hash table elements across memory into a dense alignment for better spatial locality.

We reorganize hash table as a hierarchical structure and leverage caches to optimize spatial locality. The agent table constructed during the simulation is organized as three-level hierarchical tables in Fig. 2B. This hierarchical hash table (HHT) is composed of three parts: Cacheline Hash Table (CHT), Software Hash Table (SHT) and Bulk-Info Table (BIT). Agent indices and bulky data are stored separately, while extra projection index i is introduced to connect them. CHT and SHT store (k, i) pairs, in which i indicates address offset to locate d. BIT is a compact array where agents' information data ds are aligned contiguously and accessed by i transferred from former hash tables. Although projection indices possess extra memory, compact alignment of agent information data saves prominent memory occupancy. In addition, MAS computation queries the agent's existence frequently leaving its information alone, such a key-value separating architecture is expected to save memory occupancies and boosts key-only lookup speed.

CHT leverages the characteristics of cache to accelerate hash operations. Data is transferred in fixed block size between cache and main memory, namely cache line. In nowaday processors, a typical cache line size is 64-bytes. We construct cache line as the basic CHT element, where hash values derived from keys are located to address of cache line. In a line, data-level parallelism is implemented, where first 32 bytes make up 8-key group and the rest is corresponding index group. As elements within a cache line share the same hash value, collision occurs when lookup CHT. We use SIMD instructions to handle this issue and accelerate lookup and update in the line. SHT is used as a victim table, which holds pairs overflowing from CHT. It has a traditional structure and occupies subtle space with respect to CHT to maintain CHT priority.

Major interfaces of HHT contain three hash-table operations: find, insert and delete. Find operator decomposes key-value querying requestion into two parts. CHT has the priority to search for the key. If CHT's lookup is not resolved, i.e., the key is not found and the line is full, it continues searching across SHT. Key searching operation returns corresponding projection address i of k. If i is meaningful and bulky data is required, specific values are extracted as located at i in BIT. For implementing insert and delete operations, two extra labels are introduced representing their states. Insert operator

fills content at empty seat of the line in CHT, and turns to SHT if a cache-line overflow occurs. Storage of the bulky value makes continuously increments at the end of BIT. Moreover, delete operation is achieved by replacing target content with the delete label. As a result, CHT and BIT make dense alignment for key-value storage and can reduce overhead for accessing cache.

Results and discussion

The megacity Shenzhen with a population larger than ten million is adopted as a study case. We implement the optimization methods on spatial explicit MAS system without NPIs proposed by Yin [17]. In the system, 11.2 million agents with demographic characteristics are synthesized in assistance of cross-referencing census data and house-hold travel survey [58]. Agents' hourly movements are formulated from mobile phone trajectory records or house-hold travel survey. Each individual is anchored onto different types of buildings for a daily trajactery, including living, working, studying and performing other activities. The synthesized agents are modeled to contact with each other when staying at the same location within one hour.

We perform the simulation on an Intel Core i9-10850K CPU platform with 10 physical cores and 20 threads. It is configured with 64 GB main memory, and cooperates with 32 KB L1 DCache, 256 KB L2 Cache and 20 MB L3 Cache. As case-focused MAS is convenient to be implemented in data-parallel manner, we perform multi-thread computation in assistant of C++11's **thread**, **mutex** and **atomic** libraries. **thread** is used to create and detach threads for agent-parallel processing. **mutex** and **atomic** help to avoid changing memory values simultaneously among threads.

First, performances between Algorithm 1 and Algorithm 2 are present on single-thread mode. As shown in Table 1, both of these methods generate almost the same amount of total infected agents. But their time evolution diverges as demonstrated in Fig. 3. Convolution with gaussian function is implemented to eliminate divergences, and parameters are fitted with ROOT toolkit [59] as $N(6, 13^2)$. It results that average delayed time is 6 days with 13-day deviation. In the statistical perspective, 68% of cases would be

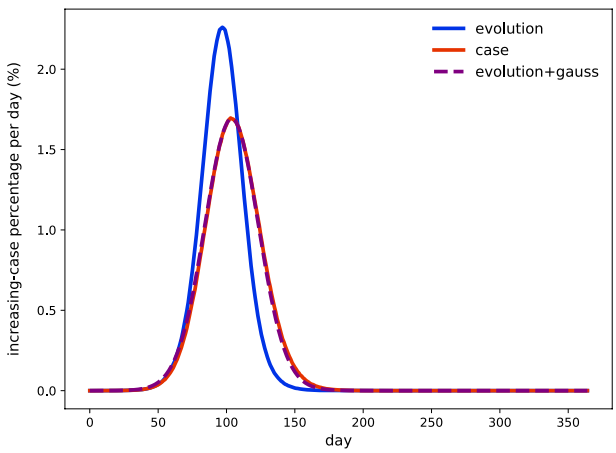

Fig. 3 Evolutions of increasing cases per day among different methods. Blue solid line is calculated by evolution-focused MAS. Read solid line is calculated by case-focused MAS. Purple dashed line is the result from convoluting evolution-focused MAS outputs with delayed-gaussian function $N(6, 13^2)$

observed with a delay of 6 ± 13 days as calculated by the case-focused MAS, and a typical confidence of 95% derives this range with 6 ± 26 days.

We perform error analysis about parameters applied in the MAS. First, both values of infectious probability and period are assumed with 20% uncertainties. Simulation results indicate that they have similar effects on the amount of total infected agents, which generate variations of the amount within $[-7\%, 4\%]$ respectively. Second, the stability of the gaussian kernel is evaluated under uncertainties of the infectious probability. We find parameters of the kernel can remain almost unchanged within its deviation $[-5\%, 20\%]$ considering statistical uncertainties, and results are shown in Fig. 4. This systematic error of Algorithm 2 originates from overlapping contact networks of susceptible individuals under different iteration orders. It will have little effects as the infected agents possess a minor amount of the population. We produce this scenario by assuming in-time quarantined of infectious population to heavily suppress the diseases. Results demonstrate that both Algorithm 1 and Algorithm 2 are equivalent to each other under statistical uncertainties.

As mentioned above, case-focused iteration has potential to significantly improve system performance by maintaining data locality. We use **perf** provided by Linux to evaluate cache accesses during simulation. As shown in Table 1, results indicate that almost 94% cache references are saved and cache misses are reduced by 90%. This optimized algorithm accelerates simulation prominently and achieves $\times 4.4$ speedup. As a result, it maintains the data locality well and improves calculation speed significantly, but sacrificing part of precision.

Next, proposed hash table HHT is implemented for further optimizing system performance under the case-focused method. As SHT is provided by system and holds victims overflowing from CHT, frequently accessing SHT may affect HHT's querying performance. We set its member occupancies is 1% to CHT capacity to minimize the influence. Algorithm 1 with libstdc++'s C++11 unordered_map under single-thread operation is adopted as benchmark. The single-thread computation is shown in the first ticks along thread axis of Fig. 5. HHT achieves $\times 6.4$ speedup which leverages both algorithm and hash table optimization. Comparing the result with unordered_map implemented under the same MAS algorithm, HHT is still faster by 47%.

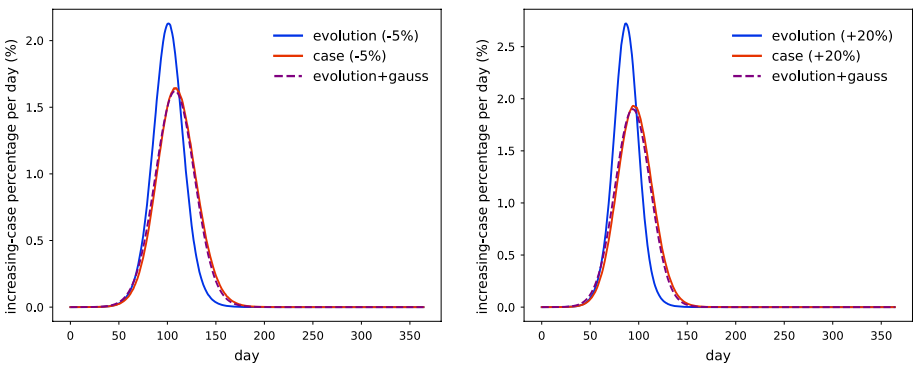

Fig. 4 Evolutions of increasing cases per day among different methods with variances of infectious probability as -5% (Left) and +20% (Right). Blue solid lines are calculated by evolution-focused MAS. Read solid lines are calculated by case-focused MAS. Purple dashed lines are the result from convoluting evolution-focused MAS outputs with delayed-gaussian function $N(6, 13^2)$

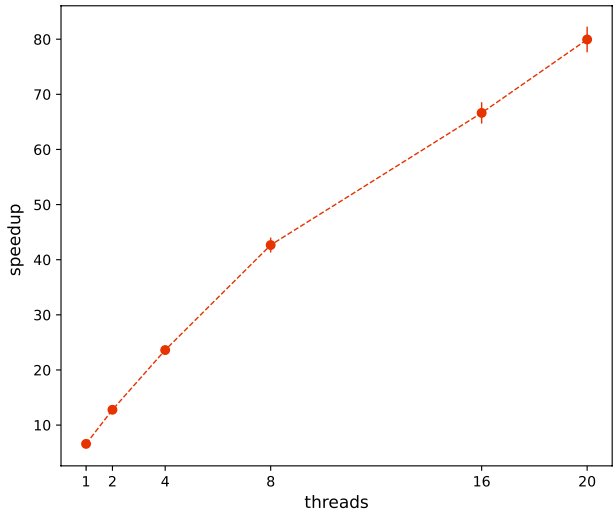

Fig. 5 Multi-thread speedup of HHT implementation on the case-focused MAS comparing to benchmark (unordered_map implementation on the evolution-focused MAS)

Table 1 Comparison between Evolution-focused MAS and case-focused MAS on single-thread mode

	Evolution-focused MAS	**Case-focused MAS**
Total Infected Agents	84%	84%
Cache References	2.5×10^{11}	1.5×10^{10}
Cache Misses	8.5×10^{10}	8×10^{9}
Running Time (s)	$2.6(\pm 0.7) \times 10^{3}$	$6(\pm 0.04) \times 10^{2}$
Speedup	$\times 1$	$\times 4.3(\pm 0.1)$

Traditional hash table is a thread-unsafe data mapping paradigm. It requires multiple memory duplication and migration during the table expands or shrinks. As hash **insert** and **find** encounter from different threads, while the **insert** causes rehashing and expands the table's capacity by factor 2, another thread may lose the actual access to target and a program fault occurs. We use atomic operator on HHT to maintain a thread-safe manner across multi-thread operations. As demonstrated by Fig. 5, this parallel implementation prominently boosts the program, where original 40-min process is compressed within $\sim$ 30s. Consequently, parallel acceleration with 20 threads achieves $\times 80(\pm 2)$ speedup to the benchmark, and processor efficiency is derived roughly 64% accordingly.

Conclusions

The MAS is an essential method for studying epidemiology sciences, especially for intervention for COVID-19 transmission. Traditional methods need querying over population as time evolves and suffers from poor-locality issues. In this work, we focus on optimizing MAS for infectious diseases, and propose two successive processes to accelerate computation. First, we reconstruct the iteration order of MAS and propose the case-focused method. It can improve data locality significantly,

where 90% cache references are saved. Consequently, the program is accelerated by ×4.3. Next, we design a thread-safe and high-performance hash table HHT for managing intermediate products from simulation. HHT leverages cache characteristics and SIMD instructions to optimize hash operations. It can further get a faster speed by 47% compared with classical hash table, and a 20-thread implementation achieves ×80 speedup finally.

It should be noticed that major MAS researches adopted evolution-focused method as simulation backbone [60–62]. As indicated by [36], general MAS methods for infectious diseases have to handle irregular and randomly accessing agent states, which is a major factor limiting the calculation efficiency. Our proposed case-focused method overcomes their disadvantages to improve data locality significantly, and this method is suitable not only for COVID-19 but also for other infectious diseases. However, it takes a trade-off with blurred and delayed results. Although the results can be recovered by deconvoluting from a mixing kernel, kernel parameters need pre-calibrated under a specific scenario. Careful studies for this mechanism will be implemented in next works. Moreover, NPI and PI methods are crucial for intervening spreading of infectious diseases. Studies of fast calculation with these interventions are left in the future.

Abbreviations

MAS	Multi-agent simulation
COVID-19	Coronavirus disease 2019
HHT	Hierarchical hash table
CHT	Cacheline hash table
SHT	Software hash table
BIT	Bulk-info table
CPU	Central processing unit
GPU	Graphics processing unit
FPGA	Field programmable gate array
SIMD	single instruction, multiple data
CDC	Center for disease control
SLIR	susceptible-latent-infectious-removed

Acknowledgements
We thank local CDC collaborators in Shenzhen for data support in this work.

Author Contributions
CJ contributed this study and wrote the manuscript. CJ, HZ and LY developed the model. YZ and SF reviewed this manuscript. All authors read and approved the manuscript.

Funding
National Key Research and Development Program of China (Grant No. 2020YFB0204802)

Availability of data and materials
Mobile phone data were provided by the Shenzhen Transportation Operation Command Center (Contact: Binliang Li, 240854198@qq.com). Travel survey data, building survey data and census data were offered by the Planning and Natural Resources Bureau of Shenzhen Municipality (Contact: Renrong Jiang, jiangrenrong@126.com). The epidemic surveillance data were provided by the Shenzhen Center for Disease Control and Prevention (Contact: Shujiang Mei, sjmei66@163.com). Researchers who meet the criteria for accessing to confidential data can send requests to the above local government departments. The daily confirmed cases of COVID-19 are publicly accessible from the Shenzhen Municipal Health Commission (http://wjw.sz.gov.cn/yqxx/). Baidu migration data can be openly obtained from http://qianxi.baidu.com/.

Declarations

Ethics approval and consent to participate
Not applicable

Competing interests
The authors declare that they have no competing interests.

Consent for publication
Not applicable

Authors' information
Not applicable

Received: 20 February 2022 Accepted: 15 June 2022
Published online: 01 July 2022

References

1. Sansores C, Pavón J. Agent-based simulation replication: a model driven architecture approach. In: Mexican international conference on artificial intelligence, 2005;244–53 . Springer.
2. Davidsson P. Agent based social simulation: a computer science view. J Artif Soc Soc Simul. 2002;5(1).
3. Wolf S, Fürst S, Mandel A, Lass W, Lincke D, Pablo-Marti F, Jaeger C. A multi-agent model of several economic regions. Environ Model Softw. 2013;44:25–43.
4. Chen D. Modeling the spread of infectious diseases: a review. Analyzing and modeling spatial and temporal dynamics of infectious diseases, 2014;19–42.
5. Domínguez R, Cannella S, Framinan JM. Scope: a multi-agent system tool for supply chain network analysis. In: IEEE EUROCON 2015-international conference on computer as a tool (EUROCON), 2015;1–5, IEEE.
6. McArthur SD, Davidson EM, Catterson VM, Dimeas AL, Hatziargyriou ND, Ponci F, Funabashi T. Multi-agent systems for power engineering applications-part i: concepts, approaches, and technical challenges. IEEE Trans Power Syst. 2007;22(4):1743–52.
7. Wangapisit O, Taniguchi E, Teo JS, Qureshi AG. Multi-agent systems modelling for evaluating joint delivery systems. Procedia Soc Behav Sci. 2014;125:472–83.
8. Dorri A, Kanhere SS, Jurdak R. Multi-agent systems: a survey. IEEE Access. 2018;6:28573–93.
9. Kermack WO, McKendrick AG. A contribution to the mathematical theory of epidemics. Proceedings of the royal society of london Series A, Containing papers of a mathematical and physical character. 1927;115(772):700–21.
10. Bailey NT. The mathematical theory of infectious diseases and its applications. In: The mathematical theory of infectious diseases and its applications, 1975;413–413.
11. Dion E, VanSchalkwyk L, Lambin EF. The landscape epidemiology of foot-and-mouth disease in South Africa: a spatially explicit multi-agent simulation. Ecol Model. 2011;222(13):2059–72.
12. Yergens D, Hiner J, Denzinger J, Noseworthy T. Multiagent simulation system for rapidly developing infectious disease models in developing countries. In: Proceedings of the 2nd international workshop on multi-agent systems for medicine and computational biology, Hakodate, Japan, 2006;104–116.
13. Castro BM, de Melo YdA, Dos Santos NF, da Costa Barcellos AL, Choren R, Salles RM. Multi-agent simulation model for the evaluation of covid-19 transmission. Comput Biol Med. 2021;136: 104645.
14. Vyklyuk Y, Manylich M, Škoda M, Radovanović MM, Petrović MD. Modeling and analysis of different scenarios for the spread of covid-19 by using the modified multi-agent systems-evidence from the selected countries. Res Phys. 2021;20: 103662.
15. Nanna GA, Quatraro NF, De Carolis B. A multi-agent system for simulating the spread of a contagious disease. In: WOA, 2020:1613;119.
16. Lorig F, Johansson E, Davidsson P. Agent-based social simulation of the covid-19 pandemic: a systematic review. JASSS: J Artif Soc Soc Simul 2021:24(3).
17. Yin L, Zhang H, Li Y, Liu K, Chen T, Luo W, Lai S, Li Y, Tang X, Ning L, et al. A data driven agent-based model that recommends non-pharmaceutical interventions to suppress coronavirus disease 2019 resurgence in megacities. J R Soc Interface. 2021;18(181):20210112.
18. Zhou S, Zhou S, Zheng Z, Lu J. Optimizing spatial allocation of covid-19 vaccine by agent-based spatiotemporal simulations. GeoHealth. 2021;5(6):000427.
19. Eubank S, Guclu H, Kumar VA, Marathe MV, Srinivasan A, Toroczkai Z, Wang N. Modelling disease outbreaks in realistic urban social networks. Nature. 2004;429(6988):180–4.
20. Ferguson NM, Cummings DA, Cauchemez S, Fraser C, Riley S, Meeyai A, Iamsirithaworn S, Burke DS. Strategies for containing an emerging influenza pandemic in southeast asia. Nature. 2005;437(7056):209–14.
21. Germann TC, Kadau K, Longini IM, Macken CA. Mitigation strategies for pandemic influenza in the united states. Proc Natl Acad Sci. 2006;103(15):5935–40.
22. Barceló J, Ferrer JL, García D, Florian M, Le Saux E. Parallelization of microscopic traffic simulation for att systems analysis. In: Equilibrium and Advanced Transportation Modelling, pp. 1998;1–26. Springer.
23. Aydt H, Xu Y, Lees M, Knoll A. A multi-threaded execution model for the agent-based semsim traffic simulation. In: Asian Simulation Conference, 2013;1–12 . Springer.
24. Saprykin A, Chokani N, Abhari RS. Large-scale multi-agent mobility simulations on a gpu: towards high performance and scalability. Proc Comput Sci. 2019;151:733–8.
25. Rao DM. Accelerating parallel agent-based epidemiological simulations. In: Proceedings of the 2nd ACM SIGSIM Conference on Principles of Advanced Discrete Simulation, 2014;127–38.
26. Cameron GD, Duncan GI. Paramics-parallel microscopic simulation of road traffic. J Supercomput. 1996;10(1):25–53.
27. Rickert M, Nagel K. Dynamic traffic assignment on parallel computers in transims. Futur Gener Comput Syst. 2001;17(5):637–48.
28. Luke S, Cioffi-Revilla C, Panait L, Sullivan K. Mason: a new multi-agent simulation toolkit. In: Proceedings of the 2004 Swarmfest Workshop, 2004;8:316–27 . Michigan, USA.
29. Amouroux E, Chu T-Q, Boucher A, Drogoul A. Gama: an environment for implementing and running spatially explicit multi-agent simulations. In: Pacific Rim International Conference on Multi-Agents, 2007;359–371 . Springer.

30. North MJ, Collier NT, Ozik J, Tatara ER, Macal CM, Bragen M, Sydelko P. Complex adaptive systems modeling with repast simphony. Complex Adapt Syst Model. 2013;1(1):1–26.
31. Cui L, Chen J, Hu Y, Xiong J, Feng Z, He L. Acceleration of multi-agent simulation on fpgas. In: 2011 21st international conference on field programmable logic and applications, 2011;470–473 . IEEE.
32. Zhou X, Fu W. A multi-agent simulation method of urban land layout structure based on fpga. Mob Netw Appl. 2020;25(4):1572–81.
33. Jang M-W, Agha G. Dynamic agent allocation for large-scale multi-agent applications. A parametric model for large scale agent systems, 2005;252.
34. Jang M-W, Agha G. Adaptive agent allocation for massively multi-agent applications. In: International workshop on massively multiagent systems, 2004;25–39 . Springer.
35. Willem L, Stijven S, Tijskens E, Beutels P, Hens N, Broeckhove J. Optimizing agent-based transmission models for infectious diseases. BMC Bioinform. 2015;16(1):1–10.
36. Zou P, Lü Y-S, Wu L-D, Chen L-L, Yao Y-P. Epidemic simulation of a large-scale social contact network on gpu clusters. Simulation. 2013;89(10):1154–72.
37. Aleta A, Martin-Corral D, Pastore y Piontti A, Ajelli M, Litvinova M, Chinazzi M, Dean NE, Halloran ME, Longini IM Jr, Merler S, et al. Modelling the impact of testing, contact tracing and household quarantine on second waves of covid-19. Nat Hum Behav. 2020;4(9):964–71.
38. Davies NG, Kucharski AJ, Eggo RM, Gimma A, Edmunds WJ, Jombart T, O'Reilly K, Endo A, Hellewell J, Nightingale ES, et al. Effects of non-pharmaceutical interventions on covid-19 cases, deaths, and demand for hospital services in the uk: a modelling study. The Lancet Public Health. 2020;5(7):375–85.
39. Li R, Pei S, Chen B, Song Y, Zhang T, Yang W, Shaman J. Substantial undocumented infection facilitates the rapid dissemination of novel coronavirus (sars-cov-2). Science. 2020;368(6490):489–93.
40. Li Q, Guan X, Wu P, Wang X, Zhou L, Tong Y, Ren R, Leung KS, Lau EH, Wong JY, et al. Early transmission dynamics in wuhan, china, of novel coronavirus–infected pneumonia. New Engl J Med. 2020;382:1199–207.
41. Zhou Y, Xu R, Hu D, Yue Y, Li Q, Xia J. Effects of human mobility restrictions on the spread of covid-19 in shenzhen, china: a modelling study using mobile phone data. The Lancet Digital Health. 2020;2(8):417–24.
42. Zhang J, Klepac P, Read JM, Rosello A, Wang X, Lai S, Li M, Song Y, Wei Q, Jiang H, et al. Patterns of human social contact and contact with animals in shanghai, China. Sci Rep. 2019;9(1):1–11.
43. Koo JR, Cook AR, Park M, Sun Y, Sun H, Lim JT, Tam C, Dickens BL. Interventions to mitigate early spread of sars-cov-2 in Singapore: a modelling study. Lancet Infect Dis. 2020;20(6):678–88.
44. Mizumoto K, Kagaya K, Zarebski A, Chowell G. Estimating the asymptomatic proportion of coronavirus disease 2019 (covid-19) cases on board the diamond princess cruise ship, yokohama, japan, 2020. Eurosurveillance. 2020;25(10):2000180.
45. Nishiura H, Kobayashi T, Miyama T, Suzuki A, Jung S-M, Hayashi K, Kinoshita R, Yang Y, Yuan B, Akhmetzhanov AR, et al. Estimation of the asymptomatic ratio of novel coronavirus infections (covid-19). Int J Infect Dis. 2020;94:154–5.
46. Kissler SM, Tedijanto C, Goldstein E, Grad YH, Lipsitch M. Projecting the transmission dynamics of sars-cov-2 through the postpandemic period. Science. 2020;368(6493):860–8.
47. Ferguson NM, Laydon D, Nedjati-Gilani G, Imai N, Ainslie K, Baguelin M, Bhatia S, Boonyasiri A, Cucunubá Z, Cuomo-Dannenburg G et al. Impact of non-pharmaceutical interventions (npis) to reduce covid-19 mortality and healthcare demand 2020.
48. Lauer SA, Grantz KH, Bi Q, Jones FK, Zheng Q, Meredith HR, Azman AS, Reich NG, Lessler J. The incubation period of coronavirus disease 2019 (covid-19) from publicly reported confirmed cases: estimation and application. Ann Intern Med. 2020;172(9):577–82.
49. Hu Z, Song C, Xu C, Jin G, Chen Y, Xu X, Ma H, Chen W, Lin Y, Zheng Y, et al. Clinical characteristics of 24 asymptomatic infections with covid-19 screened among close contacts in Nanjing, China. Sci China Life Sci. 2020;63(5):706–11.
50. Kocberber O, Grot B, Picorel J, Falsafi B, Lim K, Ranganathan P. Meet the walkers accelerating index traversals for in-memory databases. In: 2013 46th Annual IEEE/ACM international symposium on microarchitecture (MICRO), 2013;468–79 . IEEE.
51. Song H, Dharmapurikar S, Turner J, Lockwood J. Fast hash table lookup using extended bloom filter: an aid to network processing. ACM SIGCOMM Comput Commun Rev. 2005;35(4):181–92.
52. Melsted P, Pritchard JK. Efficient counting of k-mers in dna sequences using a bloom filter. BMC Bioinform. 2011;12(1):1–7.
53. Zhang G, Sanchez D. Leveraging caches to accelerate hash tables and memoization. In: Proceedings of the 52nd annual IEEE/ACM international symposium on microarchitecture, 2019;440–52.
54. Hayes T, Palomar O, Unsal O, Cristal A, Valero M. Vector extensions for decision support dbms acceleration. In: 2012 45th annual IEEE/ACM international symposium on microarchitecture, 2012;166–76 . IEEE
55. Lloyd S, Gokhale M. Near memory key/value lookup acceleration. In: Proceedings of the international symposium on memory systems, 2017;26–33.
56. Tanaka S, Kozyrakis C. High performance hardware-accelerated flash key-value store. In: The 2014 Non-volatile Memories Workshop (NVMW) 2014.
57. Xu S, et al. Bluecache: a scalable distributed flash-based key-value store. PhD thesis, Massachusetts Institute of Technology 2016.
58. Zhao Z, Shaw S-L, Yin L, Fang Z, Yang X, Zhang F, Wu S. The effect of temporal sampling intervals on typical human mobility indicators obtained from mobile phone location data. Int J Geogr Inf Sci. 2019;33(7):1471–95.
59. Antcheva I, Ballintijn M, Bellenot B, Biskup M, Brun R, Buncic N, Canal P, Casadei D, Couet O, Fine V, et al. Root-a c++ framework for petabyte data storage, statistical analysis and visualization. Comput Phys Commun. 2011;182(6):1384–5.
60. Cuevas E. An agent-based model to evaluate the covid-19 transmission risks in facilities. Comput Biol Med. 2020;121: 103827.

61. Jalayer M, Orsenigo C, Vercellis C. Cov-abm: a stochastic discrete-event agent-based framework to simulate spatiotemporal dynamics of covid-19. arXiv preprint arXiv:2007.13231 2020.
62. Kou L, Wang X, Li Y, Guo X, Zhang H. A multi-scale agent-based model of infectious disease transmission to assess the impact of vaccination and non-pharmaceutical interventions: the covid-19 case. J Saf Sci Resil. 2021;2(4):199–207.

Publisher's Note