bộ nhớ khi sắp xếp hơn 10 triệu phần tử. Trong Hyperledger, hợp đồng thông minh được biên dịch và chạy trực tiếp trên máy địa phương trong môi trường Docker, do đó nó không có chi phí chung liên quan đến việc thực hiện EVM byte code ở cấp cao. Kết quả là, Hyperledger đạt hiệu quả hơn rất nhiều về tốc độ và sử dụng bộ nhớ. Cuối cùng chúng ta lưu ý rằng cả ba hệ thống không sử dụng đa lõi kiến trúc, tức là họ thực hiện các hợp đồng chỉ sử dụng một cốt lõi.

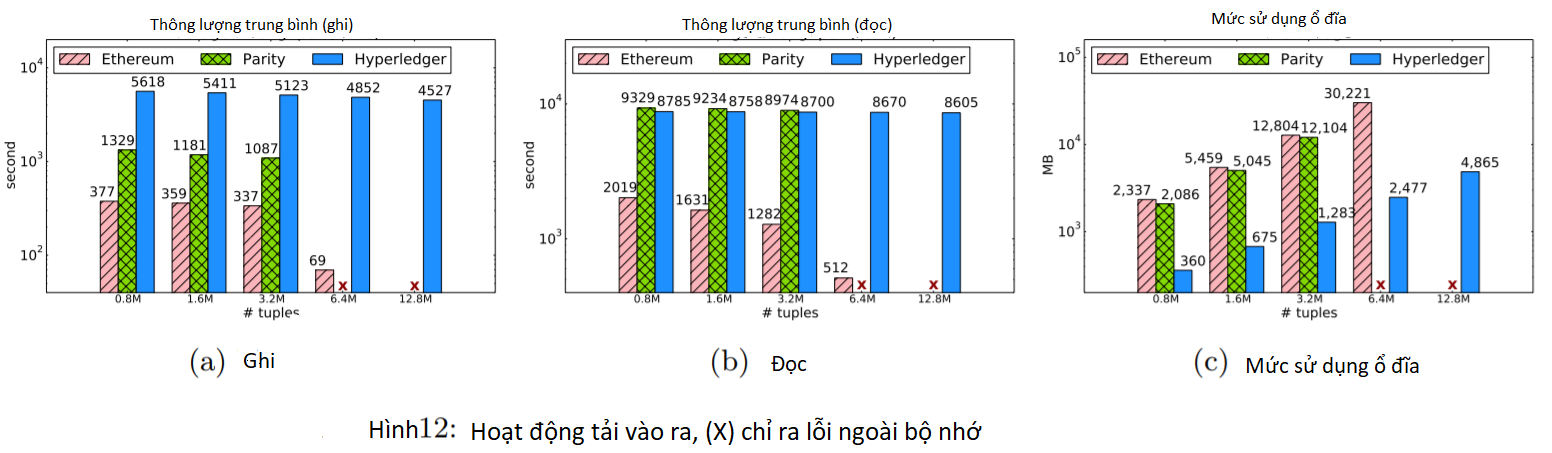
**4.2.2. Mô hình dữ liệu**

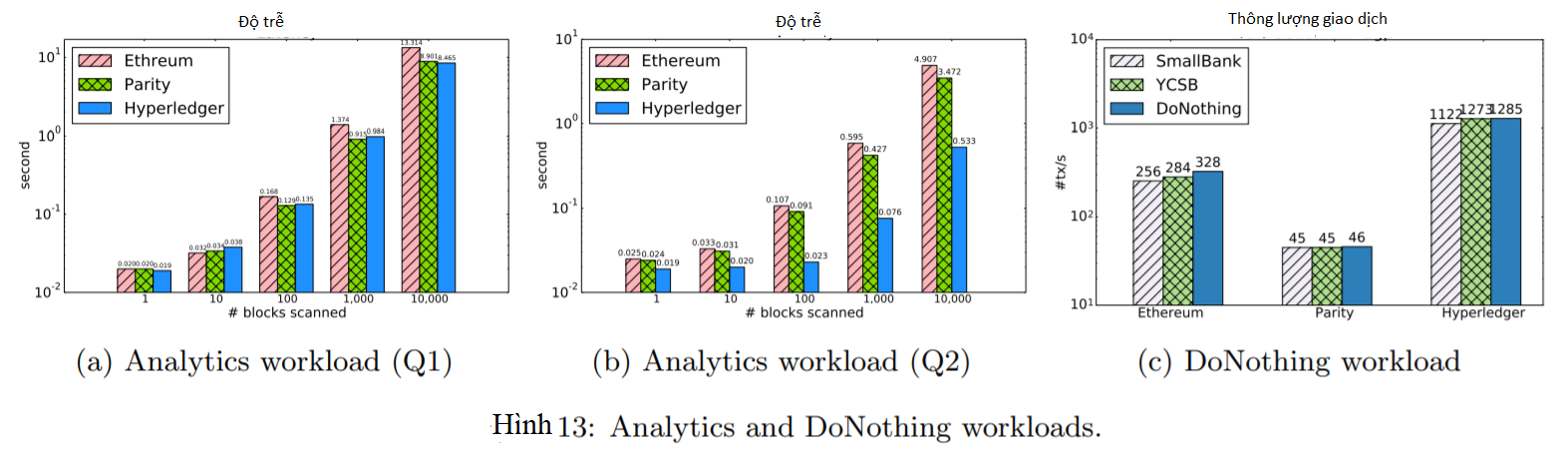
**IO Heavy**. Chúng tôi triển khai hợp đồng thông minh IOHeavy để thực hiện một lượng lớn thao tác đọc và ghi với các cặp khóa-giá trị. Chúng tôi sử dụng 20 byte cho các khóa và 100 byte cho các giá trị. Hình 12 báo cáo mức thông lượng và mức sử dụng đĩa cho những hoạt động này. Ethereum và Parity sử dụng cùng một mô hình dữ liệu và cấu trúc chỉ mục, do đó chúng tương tự nhau về chi phí không gian. Cả hai đều sử dụng yêu cầu lớn hơn nhiều về không gian lưu trữ so với Hyperledger,sử dụng mô hình dữ liệu đơn giản khóa-giá trị.Parity giữ tất cả trạng thái thông tin trong bộ nhớ, vì vậy nó có hiệu suất vào ra tốt hơn nhưng không thể xử lý dữ liệu lớn (giới hạn không quá 3 triệu trạng thái theo sự thiết lập của phần cứng). Ngược lại, Ethereum chỉ lưu trữ chỉ các bộ phận của trạng thái trong bộ nhớ (sử dụng LRU cho chính sách trục xuất), do đó nó có thể xử lý nhiều dữ liệu hơn Parity với cùng chi phí thông lượng. Hyperledger thúc đẩy RocksDB để quản lý trạng thái lưu trữ, làm cho nó hiệu quả hơn về quy mô.

**Phân tích câu truy vấn.** Chúng tôi cài đặt công việc phân tích bằng cách khởi tạo ba hệ thống với hơn 120.000 tài khoản với một sự cân bằng cố định. Sau đó, chúng tôi đã tải chúng lên 100, 000 khối, mỗi khối gồm 3 giao dịch mức trung bình. Giao dịch chuyển giá trị từ một tài khoản ngẫu nhiên đến một tài khoản ngẫu nhiên khác. Do chi phí chung của Parity trong ký kết giao dịch khi có nhiều tài khoản, chúng tôi đã xem xét giao dịch chỉ sử dụng 1024 tài khoản. Chúng tôi sau đó thực hiện hai truy vấn được mô tả trong phần 3.4 và đo độ trễ của chúng. Hình 13 cho thấy hiệu suất của Q1 là tương tự, trong khi Q2 thấy một khoảng cách đáng kể giữa Hyperledger và phần còn lại. Chúng tôi lưu ý rằng nút cổ chai chính cho cả Q1 và Q2 là số yêu cầu mạng (RPC) được gửi bởi khách hàng. Đối với Q 1, khách hàng gửi cùng một số yêu cầu cho tất cả các hệ thống, do đó hiệu suất của chúng là tương tự. Mặt khác, trong Q2, khách hàng gửi một RPC mỗi khối để Ethereum và Parity, nhưng chỉ có một RPC cho Hyperledger vì hợp đồng thông minh của chúng tôi thực hiện (xem Phụ lục C). Điều này tiết kiệm trong mạng thời gian bay khứ hồi được chuyển đến hơn 10 lần cải thiện trong độ trễ Q2 .

**4.2.3 Đồng thuận**

Chúng tôi triển khai hợp đồng thông minh DoNothing, nó chấp nhận một giao dịch và trả về ngay lập tức. Chúng tôi đã đo thông lượng tác vụ này và so sánh với YCSB và Smallbank. Sự khác biệt của các tác vụ ,được thể hiện trong hình 13 [c], là chỉ số của chi phí của giao thức đồng thuận so với phần còn lại của phần mềm ngăn xếp. Đặc biệt, đối với Ethereum chúng tôi quan sát thấy tăng 10% thông lượng so với YCSB, có nghĩa là thực hiện của tài khoản giao dịch YCSB cho phí 10%. Chúng tôi không nhận thấy sự khác biệt giữa các tác vụ trong Parity, bởi vì nút cổ chai trong Parity là do ký kết giao dịch (thậm chí các giao dịch rỗng vẫn cần phải được ký), không do sự đồng thuận hoặc thực hiện giao dịch.





**5. THẢO LUẬN**

**Hiểu biết về các hệ thống blockchain**. Khung của chúng tôi là được thiết kế để cung cấp sự hiểu biết tốt hơn về hiệu suất và thiết kế của các hệ thống blockchain riêng biệt khác. Như nhiều hệ thống blockchain đang được đề xuất, mỗi đề xuất sẽ thiết lập các bộ tính năng khác nhau, giá trị chính của Blockbench là nó thu hẹp khoảng không thiết kế thành bốn lớp trừu tượng. Khảo sát của chúng tôi về các hệ thống blockchain hiện tại (xem Phụ lục A) cho thấy bốn lớp là phù hợp có khả năng nắm bắt các đặc điểm chính của các hệ thống này. Bởi so sánh các lớp này, người ta có thể hiểu sâu hơn về thiết kế thương mại và hiệu suất tắc nghẽn. Trong trang này, ví dụ, bằng cách chạy khối lượng công việc IOHeavy chúng tôi xác định hiệu suất Parity cho khả năng mở rộng bằng cách giữ các trạng thái trong bộ nhớ. Một ví dụ khác là tính cân bằng trong mô hình dữ liệu được thực hiện bởi Hyperledger. Một mặt,mô hình đơn giản khóa-giá trị có nghĩa là một số phân tích có thể không được hỗ trợ trực tiếp. Mặt khác, nó cho phép tối ưu hoá giúp trả lời các câu hỏi hiệu quả hơn. Cuối cùng, chúng tôi xác định rằng nút cổ chai trong Parity không phải là do với giao thức đồng thuận, nhưng là do giao dịch của máy chủ ký kết. Chúng tôi cho rằng những hiểu biết như vậy không dễ trích xuất mà không có khuôn khổ phân tích hệ thống.

**Khả năng sử dụng blockchain**. Kinh nghiệm của chúng tôi trong việc làm việc với ba hệ thống blockchain khẳng định niềm tin rằng trong blockchain trạng thái hiện tại vẫn chưa sẵn sàng cho việc sử dụng hàng loạt. Cả những bản thiết kết và cơ sở mã nguồn vẫn đang được tinh chỉnh liên tục, và không có ứng dụng khác được thiết lập bên cạnh crypto-currentcy (một công nghệ liên quan tới Bitcoin). Trong ba hệ thống, Ethereum là trưởng thành hơn cả về cơ sở mã nguồn, cơ sở người dùng và nhà phát triển cộng đồng. Một vấn đề về khả năng sử dụng khác mà chúng tôi gặp phải đang chuyển các hợp đồng thông minh từ một hệ thống sang một hệ thống khác, bởi vì các mô hình lập trình riêng biệt của họ (xem Phần 3). Điều này có thể sẽ trở nên trầm trọng hơn khi có nhiều nền tảng blockchain được đề xuất [44, 16].

Mang thiết kế cơ sở dữ liệu vào blockchain. Các thách thức trong blockchain mở rộng bằng cách cải thiện các giao thức đồng thuận của nó đang được giải quyết trong nhiều sản phẩm gần đây [34, 37]. Tuy nhiên, như chúng ta đã trình bày ở phần trước, đó là vấn đề về tắc nghẽn cổ chai. Chúng tôi đề xuất bốn cách tiếp cận trong việc áp dụng các nguyên tắc thiết kế từ các hệ thống cơ sở dữ liệu để cải tiến blockchain.

*Giải phóng bộ nhớ, công cụ thực hiện và lớp đồng thuận từ mỗi khác, sau đó tối ưu hóa và quy mô chúng một cách độc lập*. Ví dụ, các hệ thống hiện tại sử dụng chung bộ lưu trữ với cặp khóa-giá trị, có thể không phù hợp nhất với cấu trúc dữ liệu duy nhất và hoạt động trong blockchain. USTore [19] thể hiện rằng một lưu trữ được thiết kế xung quanh cấu trúc dữ liệu blockchain có thể đạt được hiệu suất tốt hơn so với hiện thực.

*Nắm bắt các điểm mới của phần cứng nguyên thủy*. Nhiều hệ thống xử lí dữ liệu đang tận dụng phần cứng mới để tăng cường hiệu suất [47, 50, 21]. Đối với blockchain, việc sử dụng phần cứng đáng tin cậy, các giao thức bỏ qua lỗi Byzantine cơ bản có thể được sửa đổi để có ít tin nhắn mạng hơn [12]. Hệ thống như Parity và Ethereum có thể tận dụng đa lõi CPU và bộ nhớ lớn để cải tiến thực hiện hợp đồng và hiệu năng I / O.

*Phân đoạn*. Blockchain cơ bản là một máy nhà nước nhân rộng, trong đó mỗi nút duy trì cùng một dữ liệu. Như vậy, blockchains về cơ bản là khác với cơ sở dữ liệu hệ thống như H-Store trong đó dữ liệu được phân chia trên các nút.Việc phân đoạn giúp giảm chi phí tính toán và có thể làm cho xử lý giao dịch nhanh hơn. Thách thức chính với sự phân đoạn là đảm bảo sự nhất quán trong các đoạn. Tuy nhiên, tính nhất quán của các giao thức hiện tại được sử dụng trong các hệ thống cơ sở dữ liệu không hoạt động trong Byzantine là một khuyết điểm. Tuy nhiên, thiết kế của họ có thể cung cấp thông tin chi tiết để thực hiện một giao thức phân đoạn có thể mở rộng hơn cho blockchain. Nghiên cứu gần đây [37] đã chứng minh tính khả thi của phân đoạn các giao thức đồng thuận, làm cho các bước quan trọng hướng tới phân vùng toàn bộ blockchain.

*Hỗ trợ ngôn ngữ khai báo*. Có một bộ cấp cao các hoạt động có thể được tạo thành theo cách khai báo giúp bạn dễ dàng xác định các hợp đồng phức tạp. Nó cũng mở ra cơ hội tối ưu hóa ở cấp độ thấp để tăng tốc độ thực hiện hợp đồng.

**6. HOẠT ĐỘNG LIÊN QUAN**

Nghiên cứu về hiệu suất của các hệ thống blockchain cho đến nay được giới hạn trong các blockchain công khai. Ví dụ, [17, 15] phân tích ảnh hưởng của kích cỡ khối và thời gian lan truyền mạng trên tổng lượng thông tin. Các đề xuất cải tiến gần đây để nâng cao hiệu suất Bitcoin [27, 34, 37, 25, 43] chủ yếu tập trung vào lớp đồng thuận, trong đó các mô hình phân tích hoặc mô phỏng mạng được sử dụng để xác nhận thiết kế mới. Các khía cạnh khác nhau của Ethereum, chẳng hạn như thời gian xử lý khối (để đồng bộ với các nút khác) và thời gian xử lí các giao dịch, cũng đã được chuẩn hóa [24, 23].Phân tích sử dụng Blockbench của chúng tôi khác với những hoạt động này ở đó nó là thành phần đầu tiên đánh giá các hệ thống blockchain riêng rẽ ở quy mô lớn đối với các tác vụ từ cơ sở dữ liệu. Hơn nữa, nó so sánh hai các hệ thống khác nhau và phân tích cách thiết kế của chúng ảnh hưởng đến hiệu năng tổng thể. Sự mở rộng trong tương lai của Blockbench sẽ cho phép đánh giá so sánh nhiều hơn các thành phần chính trong blockchain.

Có nhiều khuôn khổ tiêu chuẩn cho việc chuẩn hóa hệ thống cơ sở dữ liệu. OLTP-Bench [18] chứa công việc tiêu chuẩn chẳng hạn như TPC-C cho các hệ thống giao dịch. YCSB [13] chứa các tác vụ trọng điểm. HiBench [32] và BigBench [28] là tính năng phân tích dữ liệu khối lượng lớn cho các hệ thống giống MapReduce. Blockbench chia sẻ cùng một thiết kế cấp cao như những khuôn khổ này, nhưng các tác vụ và trình điều khiển chính là được thiết kế đặc biệt cho các hệ thống blockchain.

**7. KẾT LUẬN**

Trong bài này, chúng tôi đề xuất khuôn khổ chuẩn hóa đầu tiên, được gọi là Blockbench, để đánh giá hệ thông blockchain riêng lẻ. Blockbench chứa các tác vụ để đo hiệu suất xử lý dữ liệu, và các tác vụ cho sự hiểu biết hiệu suất tại các lớp khác nhau của blockchain. Sử dụng Blockbench, chúng tôi tiến hành toàn diện phân tích ba hệ thống blockchain chính, cụ thể là Ethereum, Parity và Hyperledger với hai điểm chuẩn vĩ mô và bốn điểm chuẩn vi mô. Kết quả cho thấy hiện nay blockchain không thích hợp cho xử lý dữ liệu quy mô lớn. Chúng tôi đã chứng minh một vài tắc nghẽn tại các lớp khác nhau của ngăn xếp.

**Sự ghi nhận**

Chúng tôi muốn cảm ơn những người đánh giá nặc danh cho nhận xét và góp ý giúp chúng tôi cải tiến bài báo. Đặc biệt cảm ơn Hao Zhang, Loi Luu, các nhà phát triển từ Ethereum, Parity và Hyperledger giúp chúng tôi thiết lập thử nghiệm. Công việc này do Quỹ Nghiên cứu, Văn phòng Thủ tướng, Singapore, theo Chương trình Nghiên cứu Cạnh tranh (CRP Giải thưởng NRF-CRP8-2011-08).

**8. Tài liệu tham khảo**

[1] BlockBench: private blockchains benchmarking.

https://github.com/ooibc88/blockbench.

[2] Ethereum blockchain app platform.

https://www.ethereum.org/.

[3] Ibm watson iot.

http://www.ibm.com/internet-of-things.

[4] Leveldb. https://leveldb.org.

[5] Monax: The ecosystem application platform.

https://monax.io.

[6] Rocksdb. https://rocksdb.org.

[7] M. Apostolaki, A. Zohar, and L. Vanbever. Hijacking

bitcoin: Large-scale network attacks on crypto-currencies. <https://arxiv.org/abs/1605.07524>, 2016.

[8] P. Bailis, A. Fekete, M. J. Franklin, A. Ghodsi, J. M.Hellerstein, and I. Stoica. Coordination avoidance in database systems. In VLDB.

[9] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A.Kroll, and E. W. Felten. Sok: Research perspectives and challenges for bitcoin and crypto-currencies. In 2015 IEEE Symposium on Security and Privacy, pages 104–121. IEEE, 2015.

[10] M. Cahill, U. Rohm, and A. D. Fekete. Serializable isolation for snapshot databases. In SIGMOD, 2008.

[11] M. Castro and B. Liskov. Practical byzantine fault tolerance. In Proceedings of the third symposium on Operating systems design and implementation, pages 173–186. USENIX Association, 1999.

[12] B.-G. Chun, P. Maniatis, S. Shenker, and J. Kubiatowicz. Attested append-only memory: Making adversaries stick to their word. In SOSP, 2007.

[13] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears. Benchmarking cloud

serving systems with ycsb. In SoCC, 2010.

[14] J. C. Corbett and J. D. et al. Spanner: Google’s globally-distributed database. In OSDI, 2012.

[15] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels,A. Kosba, A. Miller, P. Saxena, E. Shi, and E. Gun. ¨ On scaling decentralized blockchains. In Proc. 3rd Workshop on Bitcoin and Blockchain Research, 2016.

[16] Crypti. A decentralized application platform. https://crypti.me.

[17] C. Decker and R. Wattenhofer. Information propagation in bitcoin network. In P2P, 2013.

[18] D. E. Difallah, A. Pavlo, C. Curino, and P. Cudre-Mauroux. Oltp-bench: An extensible testbed for benchmarking relational databases. In VLDB, 2013.

[19] A. Dinh, J. Wang, S. Wang, W.-N. Chin, Q. Lin, B. C. Ooi, P. Ruan, K.-L. Tan, Z. Xie, H. Zhang, and M. Zhang. UStore: a distributed storage with rich semantics. https://arxiv.org/pdf/1702.02799.pdf.

[20] J. Douceur. The sybil attack. In IPTPS, 2002.

[21] A. Dragojevic, D. Narayanan, E. B. Nightingale, M. Renzelmann, A. Shamis, A. Badam, and M. Castro. No compromises: distributed transactions with consistency, availability and performance. In SOSP, 2015.

[22] Ethcore. Parity: next generation ethereum browser. https://ethcore.io/parity.html.

[23] Ethcore. Performance analysis. https://blog.ethcore.io/performance-analysis/.

[24] Ethereum. Ethereum benchmarks. https://github.com/ethereum/wiki/wiki/Benchmarks.

[25] I. Eyal, A. E. Gencer, E. G. Sirer, and R. van Renesse.Bitcoin-ng: A scalable blockchain protocol. In NSDI, 2016.

[26] I. Eyal and E. G. Sirer. Majority is not enough: Bitcoin mining is vulnerable. In Fiancial Cryptography, 2014.

[27] A. Gervais, G. O. Karame, K. Wust, V. Glykantizis, H. Ritzdorf, and S. Capkun. On the security and performance of proof of work blockchains. https://eprint.iacr.org/2016/555.pdf.

[28] A. Ghazal, T. Rabl, M. Hu, F. Raab, M. Poess, A. Crolotte, and H.-A. Jacobsen. Bigbench: towards an industry standard benchmark for big data analytics. In SIGMOD, 2013.

[29] G. S. Group. Blockchain: putting theory into practice, 2016.

[30] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg. Eclipse attacks on Bitcoin’s peer-to-peer network. In USENIX Security, 2015.

[31] Hyperledger. Blockchain technologies for business. https://www.hyperledger.org.

[32] Intel. Hibench suite. https://github.com/intel-hadoop/HiBench.

[33] F. P. Junqueira, B. C. Reed, and M. Serafini. Zab: high-performance broadcast for primary-backup systems. In Dependable Systems and Networks, 2011.

[34] E. Kokoris-Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford. Enhancing bitcoin security and performance with strong consistency via collective signing. In USENIX Security, 2016.

[35] L. Lamport. Paxos made simple. SIGACT News, 2001.

[36] Q. Lin, P. Chang, G. Chen, B. C. Ooi, K.-L. Tan, and Z. Wang. Towards a non-2pc transaction management in distrubted database systems. In SIGMOD, 2016.

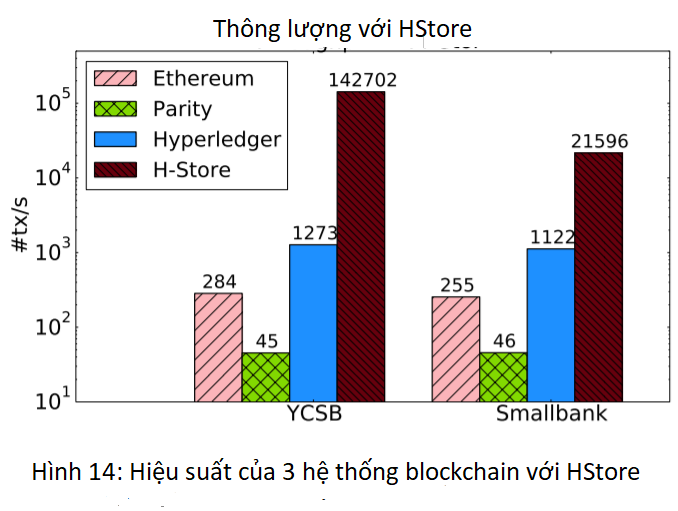
[37] L. Luu, V. Narayanan, C. Zhang, K. Baweija, S. Gilbert, and P. Saxena. A secure sharding protocol for open blockchains. In CCS, 2016.

[38] L. Luu, J. Teutsch, R. Kulkarni, and P. Saxena. Demystifying Incentives in the Consensus Computer. CCS ’15, pages 706–719, 2015.

[39] Melonport. Blockchain software for asset management. http://melonport.com.

[40] J. Morgan and O. Wyman. Unlocking economic advantage with blockchain. a guide for asset managers., 2016.

[41] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.



[42] D. Ongaro and J. Ousterhout. In search of an understandable consensus algorithm. In USENIX ATC, 2014.

[43] R. Pass and E. Shi. Hybrid consensus: efficient consensus in the permissionless model.

https://eprint.iacr.org/2016/917.pdf.

[44] Ripple. Ripple. https://ripple.com.

[45] Y. Sompolinsky and A. Zohar. Accelerating bitcoin’s transaction processing: fast money grows on trees, not chains. Cryptology ePrint Archive, Report 2013/881, 2013. https://eprint.iacr.org/2013/881.pdf.

[46] M. Stonebraker, S. Madden, D. J. Abadi, S. Harizopoulos, N. Hachem, and P. Helland. The end of and architectural era (it’s time for a complete rewrite). In VLDB, 2007.

[47] K.-L. Tan, Q. Cai, B. C. Ooi, W.-F. Wong, C. Yao, and H. Zhang. In-memory databases: Challenges and opportunities from software and hardware perspectives. SIGMOD Records, 44(2), 2015.

[48] A. Thomson, T. Diamond, S. chun Weng, K. Ren, P. Shao, and D. J. Abadi. Calvin: fast distributed transaction for partitioned database systems. In SIGMOD, 2012.

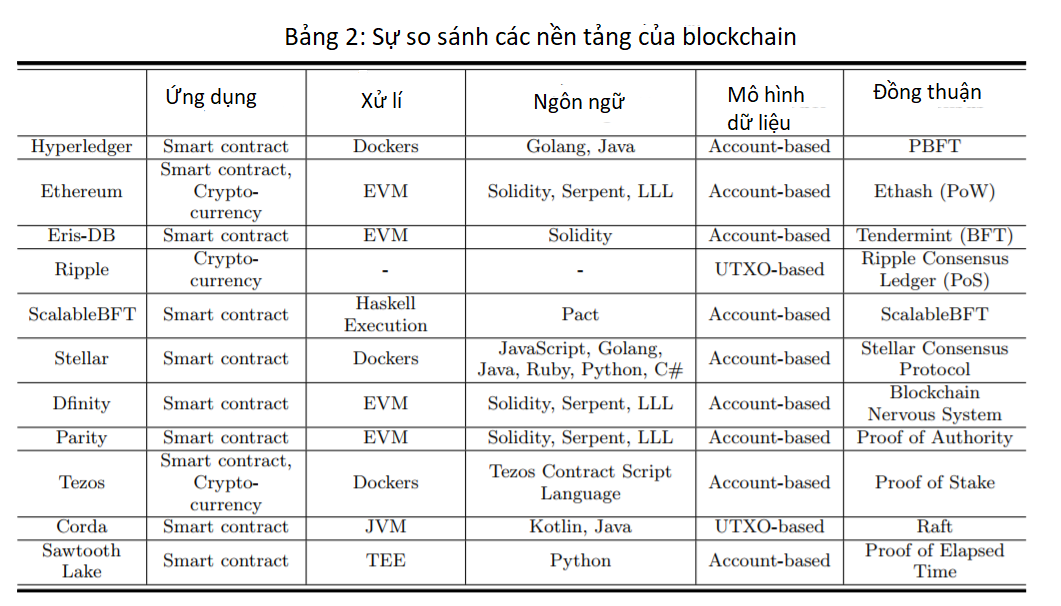
[49] M. Vukolic. The quest for scalable blockchain fabric: proof-of-work vs. bft replication. In Open Problems in Network Security - iNetSec, 2015.

[50] H. Zhang, G. Chen, B. C. Ooi, K.-L. Tan, and M. Zhang. In-memory big data management and processing: a survey. TKDE, 2015.

**Phụ lục**

**A. KHẢO SÁT CÁC NỀN TẢNG CỦA BLOCKCHAIN**

Chúng tôi so sánh mười một nền tảng blockchain đầy hứa hẹn trong Bảng 2. Chúng ta có thể thấy rằng tất cả trừ Ripple đều hỗ trợ hợp đồng thông minh. Ethereum, Eris-DB, Dfinity và Parity thực hiện hợp đồng sử dụng Ehtereum Virtual Machine-máy ảo Ehtereum- (EVM), trong khi đó Corda chạy chúng trong Java Virtual Machine-máy ảo java- (JVM). Hyperledger, Stellar và Tezos sử dụng hình ảnh Docker, ScalableBFT lấy môi trường thực hiện Haskell, và Sawtooth Lake đưa ra các hợp đồng trên đầu của Trusted Execution Enviroment-Môi trường Thực hiện Tin cậy- (TEE) như phần mềm Intel Software Guard Extensions của Intel (SGX). Những nền tảng cũng hỗ trợ các ngôn ngữ khác nhau để phát triển hợp đồng thông minh. Chẳng hạn, Solidity, Serpent và LLL chủ yếu được sử dụng trong Ethereum, Dfinity và Parity, trong khi chỉ Eris-DB hỗ trợ Solidity. Hyperledger, Stellar, Corda và Sawtooth Lake khai thác các ngôn ngữ lập trình khác nhau, chẳng hạn như Python, Java, Golang, vv ScalableBFT và Tezos thậm chí phát triển ngôn ngữ hợp đồng thông minh của riêng mình. Hầu hết các mô hình dữ liệu blockchain của nền tảng dựa trên tài khoản. Hai ngoại lệ trong bảng là Ripple và Corda. Các mô hình dữ liệu của chúng tương tự như các kết quả giao dịch không chủ ý của Bitcoin (UTXO) đại diện cho đồng tiền trong mạng.



Mỗi nền tảng cung cấp các giao thức đồng thuận khác nhau. Hyperledger thực hiện PBFT trong phiên bản chúng tôi đánh giá, trong khi Ethereum thực hiện một biến thể của PoW (Proof-ofWork – Bằng chứng hành động). Eris-DB xây dựng trên giao thức Tendermint nhưng chỉ hoạt động trong phiên bản mới nhất (v 0.12). Ripple và Tezos triển khai kế hoạch Proof-of-Stake (PoS) (một trong Ripple là được gọi là Ripple Consensus Ledger) trong đó khối tiếp theo được tạo ra dựa trên sự giàu có của tài khoản, nghĩa là cổ phần. Parity phải mất một thoả thuận khác, Proof-of-Authority (PoA), trong đó có một bộ "cơ quan" được xác định trước để tạo mới khối trong một khe thời gian cố định và bảo đảm mạng lưới blockchain. Sawtooth Lake sử dụng Bằng chứng về Thời gian Đã qua Sử (Proof-of-Elapsed-Time)(PoET) làm giao thức đồng thuận, trong tự nhiên là một thuật toán xổ số và quyết định người sáng tạo của khối một cách tùy tiện. Stellar phát triển cơ chế riêng của nó, giao thức đồng thuận Stellar, là một xây dựng cho các thỏa thuận Byzantine phân cấp. Không có mã nguồn giúp xác định giao thức đồng thuận Dfinity sử dụng, nhưng các tài liệu của nó cho thấy rằng một Blockchain là Nervous System (hệ thần kinh) sẽ điều chỉnh toàn bộ nền tảng thông qua một cơ chế bỏ phiếu dựa trên nơ-ron tương tác với mỗi khác và được kiểm soát bởi người dùng.

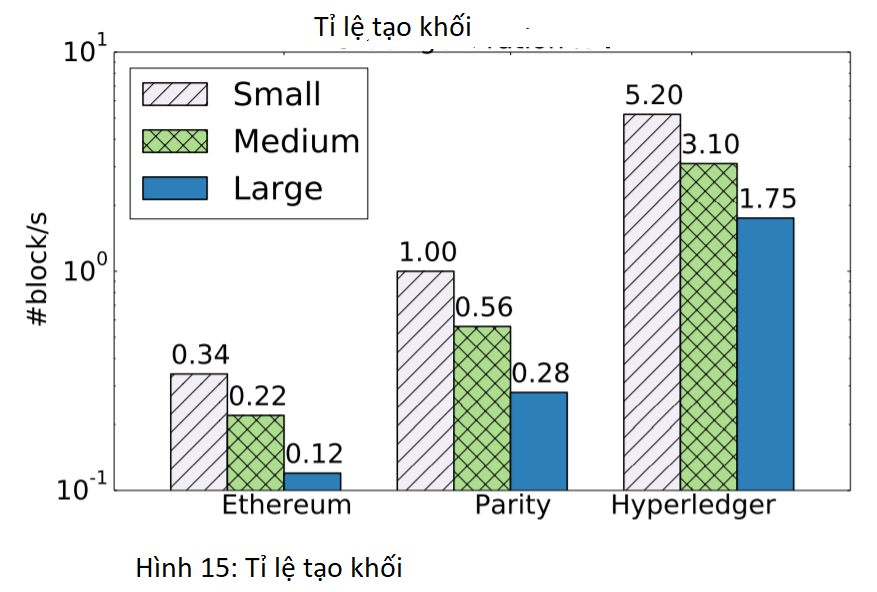
B. CÁC ĐIỂM CHUẨN HÓA VĨ MÔ

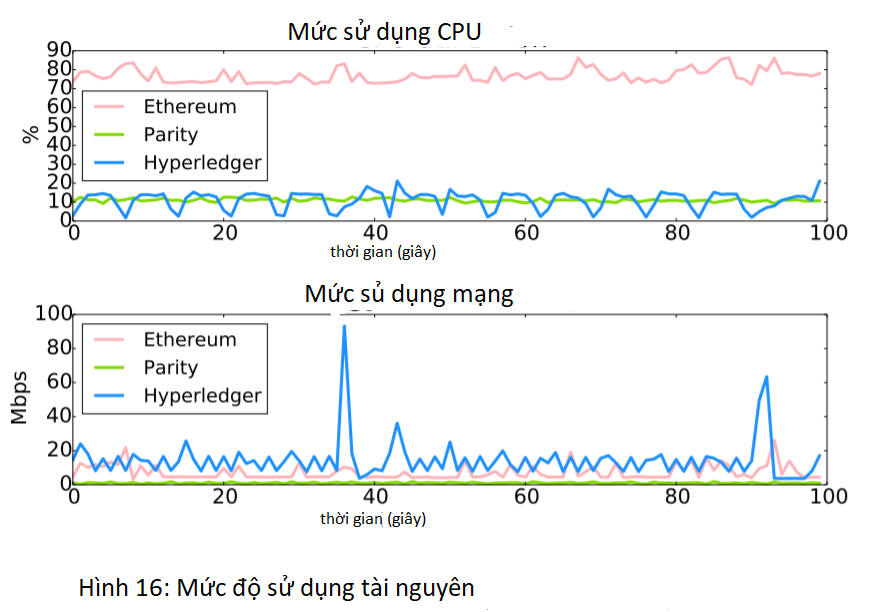
Chúng tôi so sánh hiệu suất của ba hệ thống blockchain đối với một hệ thống cơ sở dữ liệu phổ biến trong bộ nhớ, cụ thể là H-Store, sử dụng tác vụ YCSB và Smallbank. Chúng tôi đã chạy Trình điều khiển chuẩn của chính H-Store và đặt tỷ lệ giao dịch ở mức 100.000 tx / s. Hình 14 cho thấy ít nhất một thứ tự cường độ khoảng cách thông lượng và hai độ lớn trong độ trễ. Cụ thể, H-Store đạt được thông lượng trên 140.000 tx / s trong khi vẫn duy trì độ trễ dưới mili giây. Khoảng trống trong hiệu suất là do chi phí của các giao thức thống nhất. Ví dụ: YCSB, H-Store yêu cầu hầu như không có sự phối hợp trong số các tác vụ cùng mức, trong khi Ethereum và Hyperledger chịu chung chi phí của PoW và PBFT.

Một quan sát thú vị là chi phí của Smallbank. Nhớ lại rằng Smallbank là một tác vụ giao dịch phức tạp hơn hơn YCSB, trong đó nhiều khóa được cập nhật trong một một giao dịch. Smallbank l đơn giản nhưng là đại diện của một khối lượng lớn tác vụ giao dịch như TPC-C. Chúng tôi nhận thấy rằng trong H-Store, Smallbank đạt mức thấp hơn 6.6x thông lượng và độ trễ cao hơn 4 lần so với YCSB, chỉ ra chi phí của giao thức phân phối quản lý giao dịch, vì H-Store là một cơ sở dữ liệu bị phân tán. Ngược lại, blockchain chịu suy giảm hiệu suất: 10% thông lượng và 20% về độ trễ. Điều này là do mỗi nút trong blockchain duy trì toàn bộ trạng thái (máy trạng thái nhân rộng), do đó không có phí quản lý phối hợp giao dịch khi dữ liệu không được phân chia.

Các kết quả chứng minh rằng blockchain thực hiện kém tại các nhiệm vụ xử lý dữ liệu hiện đang được xử lý bởi các hệ thống cơ sở dữ liệu. Tuy nhiên, chúng tôi nhấn mạnh rằng blockchains và cơ sở dữ liệu được thiết kế với các mục tiêu và giả định khác nhau. Đặc biệt, các giao thức để bỏ qua lỗi Byzantize là một việc quá sức cho việc thiết lập các hệ cơ sở dữ liệu truyền thống ở nơi chỉ có lỗi sự cố. Các tính năng khác, chúng được lựa chọn trong hầu hết các hệ thống cơ sở dữ liệu là chữ ký mật mã trên mỗi giao dịch đơn và toàn bộ khu vực được sao chép máy trạng thái. Mặc dù cơ sở dữ liệu được thiết kế mà không có các tính năng bảo mật và bỏ qua các lỗi Byzantine, chúng tôi nhận xét rằng khoảng cách vẫn còn quá cao đối với các blockchains gây rối đến các hệ thống cơ sở dữ liệu đương nhiệm. Tuy nhiên, sự phổ biến của blockchain là một dấu hiệu rõ ràng rằng có một nhu cầu đối với một hệ thống xử lý dữ liệu Byzantine có thể chứa một số lượng lớn người dùng.

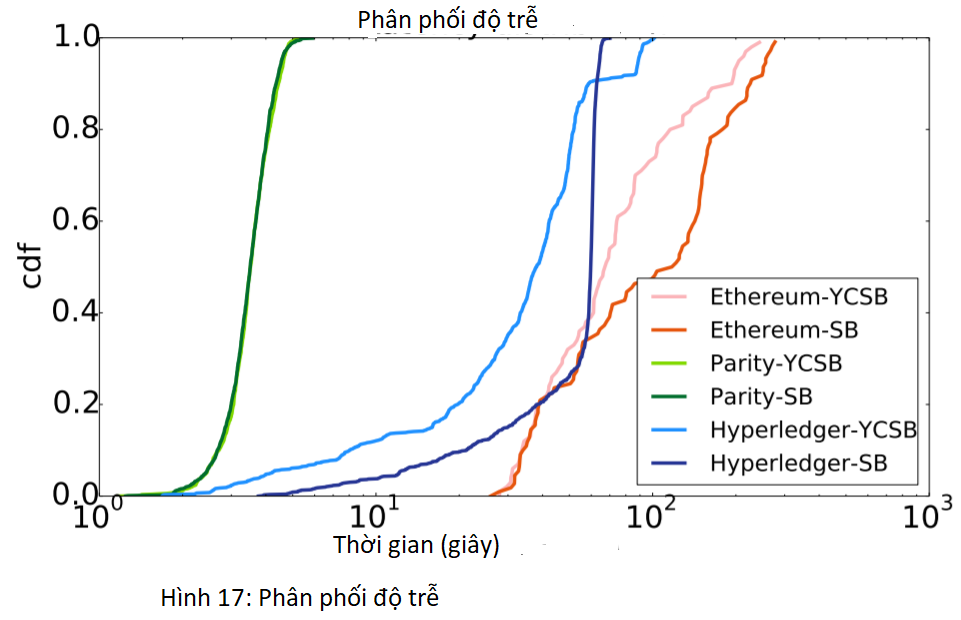
Hình 15 cho thấy hiệu quả của việc thay đổi các kích cỡ khối trong thông lượng tổng thể.Trong khi đơn giản để thiết lập kích thước một khối trong Hyperledger bằng cách cấu hình biến **batchSize**, thì không có cách trực tiếp nào để xác định điều tương tự ở Ethereum. Một người khai thác Ethereum sử dụng giá trị **gasLimit** để hạn chế chi phí tổng thể để xây dựng một khối, do đó chúng tôi đã điều chỉnh giá trị này mô phỏng các kích cỡ khác nhau. Trong Parity, **gasLimit** không áp dụng để giao dịch địa phương và nó không có ảnh hưởng đến kích thước khối. Thay vào đó, chúng tôi quan sát thấy rằng kích thước khối có thể được kiểm soát bởi điều chỉnh giá trị **stepDuration**, nó quyết định chủ yếu lượng thời gian một người kiểm chứng có thể sử dụng để xây dựng một khối. Trong các thí nghiệm, kích thước trung bình đề cập đến cài đặt mặc định, trong khi kích thước lớn và nhỏ tham khảo từ 2x đến 0.5x của kích thước mặc định. Các kết quả cho thấy rằng tăng kích thước khối dẫn đến tỷ lệ giảm tốc độ phát sinh khối, có nghĩa là tổng thể thông lượng không cải thiện.

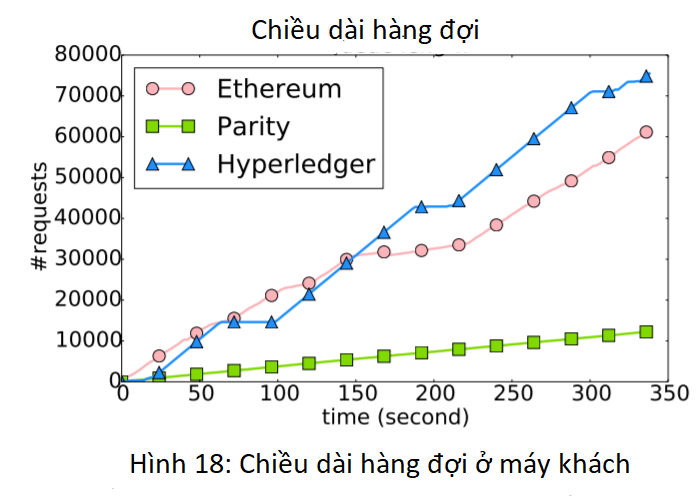


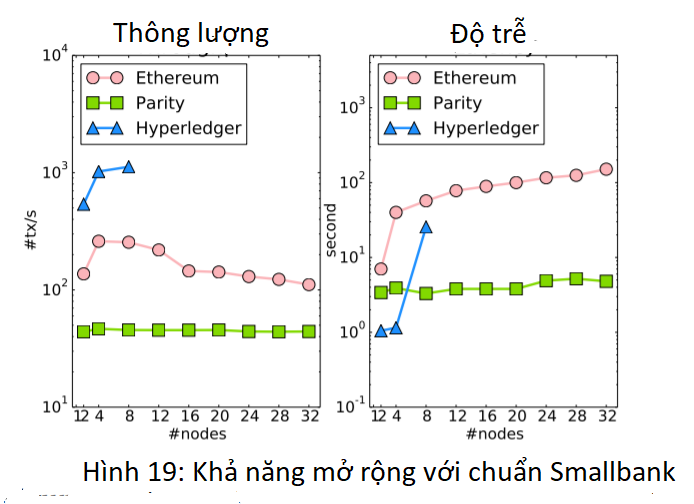


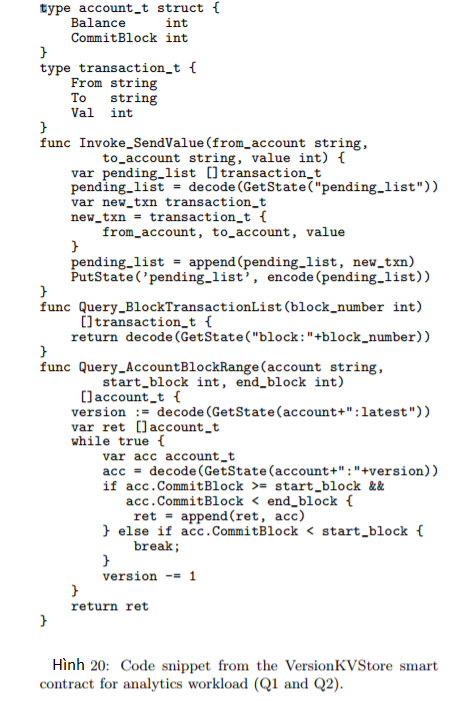
Hình 16 so sánh mức sử dụng CPU và mạng của ba hệ thống trong khoảng thời gian 100 giây. Nó dễ dàng để thấy rằng Ethereum bị CPU ràng buộc, vì nó sử dụng hoàn toàn 8 lõi CPU. Hyperledger, trên mặt khác, sử dụng CPU ít và dành phần còn lại của thời gian trên truyền thông mạng. Ngược lại, Parity có các dấu ấn tài nguyên thấp hơn hai hệ thống còn lại. Đối với Ethereum và Hyperledger, mô hình là ảnh hưởng trực tiếp đến các giao thức đồng thuận: PoW bị CPU ràng buộc trong khi PBFT bị truyền thông ràng buộc.

Hình 17 cho thấy sự phân bố độ trễ. Ethereum có cả độ trễ cao hơn và phương sai cao hơn, bởi vì PoW là một quá trình ngẫu nhiên có nghĩa là khoảng thời gian giữa khối là không thể đoán trước. Parity có sự khác biệt thấp nhất bởi vì máy chủ hạn chế tốc độ yêu cầu của khách hàng ở mức 80 tx / s.









Hình 18 minh hoạ hàng đợi yêu cầu tại máy khách các thiết lập của 20 máy chủ và 20 khách hàng. Hành vi hàng đợi của Ethereum phản ánh trường hợp bình thường, tức là hàng đợi tăng và thu hẹp tùy theo tốc độ giao dịch được thực hiện như thế nào. Hyperledger thất bại trong việc tạo ra các khối trong trường hợp này, do đó hàng đợi không bao giờ co lại. Tuy nhiên, có thời lượng trong đó kích thước hàng đợi vẫn không đổi. Hơn nữa, ở đầu, hàng đợi trong Hyperledger nhỏ hơn ở Ethereum, mặc dù khách hàng đang gửi ở mức tương tự. Điều này cho thấy có một nút cổ chai trong xử lý các yêu cầu mạng tại các máy chủ Hyperledger.

Hình 19 minh hoạ khả năng mở rộng của ba hệ thống sử dụng chuẩn của Smallbank. Chúng tôi quan sát các mẫu tương tự đến chuẩn YCSB (Hình 7), ngoại trừ Hyperledger không vượt quá 8 nút thay vì 16.

**C. PHÂN TÍCH HỢP ĐỒNG THÔNG MINH**

Hình 20 cho thấy việc thực hiện các phương thức của hợp đồng thông minh nó câu trả lời việc phân tích tác vụ của Q2. Để hỗ trợ tra cứu dữ liệu lịch sử, chúng tôi thêm một truy cập vào khóa của mỗi tài khoản. Để lấy một phiên bản cụ thể của một tài khoản, chúng tôi sử dụng khóa account:version. Chúng tôi lưu trữ phiên bản mới nhất của tài khoản sử dụng khóa account:latest, và giữ một CommitBlock trong trường dữ liệu cho mỗi phiên bản là cái cho biết khối trong bảng đối chiếu của phiên bản này là được cam kết. Để trả lời truy vấn, viêc này cần tìm danh sách đối chiếu của tài khoản được đưa ra trong một phạm vi khối nhất định, phương pháp quét tất cả các phiên bản của tài khoản này và trả lại các giá trị đối chiếu được cam kết trong phạm vi khối nhất định. Ethereum và Parity cung cấp JSON-PRC API getBalance (tài khoản, block) để truy vấn thông tin của một tài khoản tại một khối nhất định. API này chỉ lấy một phiên bản của tài khoản tại mỗi vòng lặp HTTP, vì vậy nó ít hiệu quả hơn so với việc đẩy truy vấn logic đến phía máy chủ.