Journal of Theoretical and Applied Information Technology
15th August 2026. Vol.104. No.15

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

UUC: UNIFORMLY UTILISED LAST LEVEL CACHE OF MODERN MULTICORE PROCESSORS

PURNENDU DAS[1], BISHWA RANJAN ROY[2*], BARGA DEORI[3]
[1,2*]Department of Computer Science, Assam University, Silchar, India
Department of Computer Science and Engineering, Dhemaji Engineering College, Dhemaji, India
E-mail: [1]purnendu.das@aus.ac.in, [2*]bishwa.ranjan.roy@aus.ac.in, [3]bargadeori@dec.ac.in

ABSTRACT

A set-associative last-level cache (LLC) can have free or weakly reused lines and still suffer frequent misses. The reason is simple: each address is confined to one set, whereas application footprints are rarely distributed uniformly over all sets. Existing flexible-placement caches improve this situation, but typically require extra tags, relocation chains or fixed set partnerships. This paper presents Uniformly Utilised Cache (UCC), a two-choice LLC organisation for multicore processors. Every block has a conventional home set and one tag-derived remote set. A fill is redirected only when the home set is under sustained pressure and the remote set is lightly occupied. The normal home lookup is unchanged; the remote set is searched sequentially and only when a displacement counter indicates that a remote copy may exist. Per-core insertion budgets prevent one application from occupying an excessive number of borrowed lines. At a 2 MB/core LLC budget, UCC improves single-core IPC by 6.42% on average and four-core weighted speedup by 8.50% over a conventional SRRIP cache. It outperforms modeled V-Way, ZCache, SBC, and FS-DAM configurations while requiring 2.72% metadata for a 4 MB, 16-way LLC.

Keywords: *Last-Level Cache, Multicore Processor, Cache Utilisation, Alternative Indexing, Conflict Miss.*

1. INTRODUCTION

Large on-chip caches are built from many banks and thousands of sets. Their aggregate capacity may be adequate, yet useful blocks can be evicted because a small group of sets receives a disproportionate share of the address stream [1]. The problem appears in graph analytics, pointer-heavy programs, compiler workloads, network simulation, and applications whose physical-page layout creates regular index conflicts [2]. Increasing associativity reduces these misses, but it also enlarges the tag comparison structure and raises hit energy [3], [4]. A fully associative organisation is not a practical answer for a modern LLC [5].

Several proposals relax the one-address/one-set restriction. V-Way increases tag capacity and decouples tags from data lines to approach global replacement [6]. ZCache combines skewed indexing with cuckoo-style relocation, exposing many replacement candidates without increasing the number of ways searched on a hit [7], [8]. Set Balancing Cache (SBC) associates highly used sets with underused sets [9]. Fellow-set dynamic associativity management (FS-DAM) creates groups containing normal and reserve storage and periodically adapts those groups [10]. These designs establish that nonuniform set pressure is worth addressing. Their cost, however, is not negligible: extra tag entries and pointers, multiple relocation probes, association tables, fixed pairings, or reserve-space management.

This paper explores a narrower design point. Uniformly Utilised Cache (UCC) gives each block exactly two legal locations: its ordinary home set and one deterministic remote set derived from the tag. The first lookup remains identical to a conventional LLC [11]. Remote placement is used only during fills, only for a hot-to-light set pair, and only while the requesting core remains within a small displacement budget. A remote search is issued after a home miss only when the home set's displaced-line counter is nonzero. Thus, UCC does not pay the latency or energy of two parallel probes on every access.

The mechanism grew out of an earlier dual-index cache study, but the design here is recast for a shared multicore LLC and a modern trace-driven methodology. The main contributions are:

- a two-choice placement policy that allows one pressured set to borrow capacity from several

remote sets without maintaining a dynamic association table;

- a conditional lookup path and precise counter invariants that avoid unnecessary secondary probes;
- a multicore control policy that limits remote fills per core and keeps capacity borrowing from becoming a source of unbounded interference; and
- a comparative ChampSim-style study covering 20 SPEC CPU2017 traces, ten four-core mixes, five flexible-placement organisations, sensitivity, utilisation, and metadata cost.

The algorithm is proposed in this paper with the objective to overcome the restriction of one address/one set by incorporating the remote location access for the cache line which is not found in the resident location. By allowing a cache line to be accessed from an alternative remote set when it is not available at its home set, the proposed algorithm effectively manages higher set associativity without proportionally increasing the number of tag comparisons.

The major challenge in this proposed architecture is the effective placement of the cache line into the remote location for different workloads. This approach also has a limited scope of placement freedom.

2. BACKGROUND AND MOTIVATION

A. Why LLC Capacity Becomes Standard

For a cache with N sets and W ways, an address is normally eligible for only W out of NW data lines. Let x_i denote the number of distinct live blocks mapping to set i during a reuse interval. The local overflow is on-chip caches are built from many banks

$$o_i = \max(0, x_i - W), \quad (1)$$

while the locally unused capacity is $u_i = \max(0, x_i - W)$. A conventional cache can have $\sum_i u_i > 0$ and still evict $\sum_i o_i$ useful blocks because unused ways in set j cannot serve set i. This is a placement problem, not a replacement-policy problem. SRRIP, SHiP, Hawkeye, and related policies improve victim selection within the eligible set [12], [13], [14]; they do not change the set to which a line may be allocated.

The imbalance is also time dependent. A set that is cold for one program phase may become hot later. Static hashing spreads addresses better than direct indexing, as shown by skewed-associative caches [15], but a fixed hash cannot guarantee that both candidate sets remain lightly loaded. Runtime pressure information is therefore useful, provided that the monitoring cost does not reach the critical path.

B. Position Relative to Prior Flexible Caches

Table 1 summarises the design space. V-Way offers broad replacement freedom but pays for a larger tag store and tag-to-data pointers. ZCache has a fast common-case hit but may perform several tag probes and relocations on a miss. SBC uses set association, although a fixed partner can become hot at the same time and a dynamic partner requires association state. FS-DAM adds runtime regrouping and a reserve region. UCC deliberately accepts less placement freedom than V-Way or ZCache. In exchange, it has no relocation chain, no central free list, and no partner table. Different blocks from the same home set can map to different remote sets because the remote index includes tag bits.

Table 1: Flexible-Placement LLC Organisation

Design	Candidate Space	Main State	Dominant Limitations
V-Way	Globally selected data line	Extra tags; forward/back pointers	Tag-store and indirection cost
ZCache	Many skewed candidates	Hashes and relocation state	Multi-step miss handling
SBC	Home plus associated set	Pressure and association tables	Partner may be unavailable or stale
FS-DAM	Fellow group with reserve region	Group load and regrouping state	Reserve management and movement
UCC	Home plus tag-derived remote set	Three counters/set; owner index/line	Sequential remote lookup

3. UCC ARCHITECTURE

C. A Home and Remote Sets

The home set $H(a)$ is the normal set index of block address a. For $n = \log_2 N$ set bits, the remote set is

$$R(a) = H(a) \oplus fold_n(T(a) \oplus K), \quad (2)$$

where $T(a)$ is the tag, K is a fixed per-slice seed, and $fold_n$ XOR-folds tag chunks to n bits. If the

Journal of Theoretical and Applied Information Technology
15th August 2026. Vol.104. No.15

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

result equals $H(a)$, the least significant index bit is inverted. Equation (2) is a shallow XOR network and can be evaluated while the home tag array is accessed. The seed is not a security key; it only breaks regular address correlations and can differ across LLC slices

A local line is stored in $H(a)$. A satellite line is physically stored in $R(a)$ but remains owned by $H(a)$. Each line stores an outside bit and, for a satellite, the n-bit home-set identifier. The ordinary tag is retained. The home identifier is needed because the physical index no longer reconstructs the complete block address.

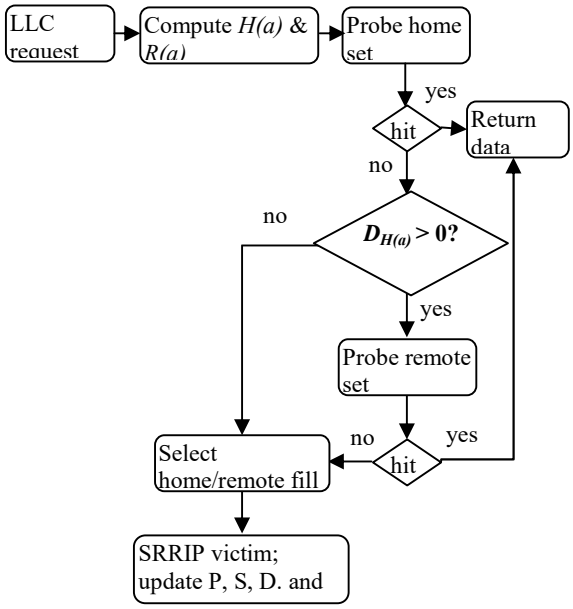

Figure 1. UCC lookup and allocation flow. The home hit path is unchanged.

D. Pressure and Occupancy State

Each set i maintains three saturating counters:

- P_i: a 6-bit pressure score, incremented on a home miss and decremented on a home hit;
- S_i: the number of satellite lines physically present in set i; and
- D_i: the number of blocks owned by set i but currently stored remotely.

For a 16-way LLC, S_i and D_i require five bits each. Scores are halved every classification epoch rather than reset, which avoids an abrupt loss of history. A set is hot when $P_i \geq \theta_H$ and light when $P_i \leq \theta_L$ and its valid occupancy does not exceed $W-2$. The default thresholds are 44 and 12. The two-way hysteresis reduces oscillation around a single boundary.

The counters have distinct meanings. S_i is physical occupancy and is consulted when considering set i as a destination. D_i is ownership state and gates remote lookup after a miss in set i. The equality

$$\sum_i S_i = \sum_i D_i \tag{3}$$

must hold after every fill, invalidation, and eviction. This simple assertion catches most implementation errors.

E. Lookup

Figure 1 shows the access sequence. The controller first probes $H(a)$ and compares only local lines (outside=0). On a home hit, data returns with the same latency as the baseline. After a home miss, $D_{H(a)}$ is read. If it is zero, no block owned by the home set is outside, so a remote lookup cannot succeed. Otherwise the controller probes $R(a)$ and compares the tag, outside bit, and stored home identifier. A remote hit updates the replacement state in the physical set. A miss in both sets proceeds to allocation.

The second lookup is sequential. Parallel probing would remove remote-hit latency, but it would access two tag arrays for every request. In the study, only 8.7% of accesses require a second probe; paying twice the tag energy on the remaining accesses is not justified.

F. Allocation and Replacement

An invalid home line is always used first. When the home set is hot, the remote set is light, and the requester has budget, the fill is sent to the remote set with probability p_r. The default $p_r = 0.7$ leaves a controlled fraction of fills local, which is useful when pressure information is stale. If these conditions are not satisfied, the home set is used. SRRIP selects the victim within the chosen physical set [12]; all compared organisations use the same underlying policy where applicable.

A remote fill sets outside=1, records the home index, increments $S_{R(a)}$ and $D_{H(a)}$, and consumes one unit of the requester's epoch budget. Evicting a satellite decrements both counters using the stored

home index. A local fill clears outside and does not change S or D. Dirty evictions follow the ordinary LLC writeback path

G. Muticore Control

Shared caches need an additional guard against capacity capture. UCC assigns each core an eight-bit remote-fill budget B_c. At the beginning of an epoch, all cores receive the same base budget. A core that generated fewer than half of its budgeted satellite fills in the preceding epoch donates the unused portion to a small common pool. A remote fill is permitted only if $B_c > 0$. This policy limits the rate at which a streaming application can displace lines in otherwise light sets, but it does not partition the LLC or prevent data sharing. The core identifier is available with the miss request and does not require a new per-line field.

The mechanism can coexist with utility-based cache partitioning [16]. A partition mask may be applied before the SRRIP victim is selected; a satellite can occupy only a way that is legal for its owner. In a NUCA LLC [17], the remote function can preserve bank bits and hash only the set-within-bank field to avoid an additional network traversal [18].

H. Storage and Timing Cost

For N sets, W ways, C cores, and $n = \log_2 N$, the extra bits are

$$O = N(q_P + q_S + q_D) + NW(n+1) + 8C, \quad (4)$$

where $q_p = 6$ and $q_S = q_D = \lceil \log_2(W+1) \rceil$. Table 2 reports two configurations. The 4 MB, 16-way point used in the cost comparison requires 112 KB, or 2.72% of data capacity. This calculation excludes baseline tags and replacement bits, which are present in every design.

The home access remains 20 cycles in the modeled shared LLC. A remote probe adds six cycles because the alternate index is ready before the home result and the request stays within the slice. The expected extra lookup delay is qL_r, where q is the second-probe rate. With $q = 0.087$ and $L_r = 6$, the average increment is 0.52 cycles per LLC access, well below a DRAM miss penalty.

Table 2: Center UUC Metadata Overhead (64-B Lines, 16 Ways)

System	Sets	Owner bits/line	Metadata	Data ovhd.
1 core, 2 MB LLC	2048	11+1	52.0 KB	2.54%
2 cores, 4 MB LLC	4096	12+1	112.0 KB	2.72%
4 cores, 8 MB LLC	8192	13+1	240.0 KB	2.93%

4. IMPLEMENTATION DETAILS

A. Control State Machine

For UUC can be added to ChampSim without changing the core or private-cache models. The LLC block structure is extended with two fields: outside and home_set. The latter is read only for satellite lines. Each LLC slice also contains the P, S, and D arrays and one budget counter per core. The remote index is computed when the request enters the LLC queue, so no hash operation is performed after the home miss becomes known.

A demand request moves through three controller states. In HOMELOOKUP, the conventional set and tag are searched. A miss with $D_{H(a)} = 0$ enters MEMORYREQUEST immediately. A miss with $D_{H(a)} > 0$ enters REMOTELOOKUP; the controller reuses the same request entry, replaces the physical set field with $R(a)$, and marks the lookup as satellite-only. A second miss then enters MEMORYREQUEST. The MSHR records both candidate sets and the requester's core until the memory response returns. This prevents the fill decision from depending on a pressure value that may have changed while the miss was outstanding.

The fill callback rechecks only physical availability, not the hot/light decision. If an invalid line observed at miss time has been consumed by another request, SRRIP chooses a victim in the previously selected destination. This rule keeps the state machine deterministic and avoids oscillation between home and remote sets. A bank conflict can delay the remote lookup, but it does not reissue the request through the on-chip network. In the modeled local-slice design, the extra operation occupies one tag-array slot and adds six cycles.

B. Atomic Counter Updates

Counter updates are tied to the physical line write. For a remote fill, the controller first identifies the victim and reconstructs its owner. If the victim is a satellite, S for the physical set and D for the old home set are decremented. The new block is then installed, its owner field is written, and the new S and D entries are incremented. The update is committed as one bank operation. A local fill performs the decrement half when necessary but does not increment displacement state.

Two requests may miss on the same home set and select the same invalid remote way. The standard ChampSim MSHR does not reserve individual ways, so the fill callback must tolerate this race. UUC handles it exactly as a conventional cache handles two fills targeting the same set: the later fill selects a current victim. The counter is therefore updated from the line actually replaced, not from the line predicted at miss time. Saturating counters protect against transient software bugs, but simulation assertions check Eq.(3) every 100,000 fills and at the end of each trace.

Coherence invalidations and writebacks require no new search structure. An invalidation first checks the home set and, when , checks the remote set. If the invalidated line is a satellite, both occupancy counters are decremented. A dirty satellite writes back the original block address formed from its tag and stored home index. Page migration or address-space remapping does not affect resident lines because the physical block address carried by the trace already determines both indices.

Table 3: UUC Metadata Overhead (64-B Lines, 16 Ways).

Step	Block	Remote	Decision	State change
1	A	214	Home invalid way	none
2	B	407	Remote fill	$S_{407}++, D_{93}++$
3	C	214	Home SRRIP victim	none
4	D	611	Remote fill	$S_{611}++, D_{93}++$
5	Read B	407	Remote hit	update SRRIP in 407
6	Evict D	611	Satellite victim	$S_{611}--, D_{93}--$

C. Worked Example

Table 3 follows four blocks whose home set is 93. Their tags produce three different remote sets. The first block uses an invalid home way. When B arrives, set 93 has crossed the hot threshold and set 407 is light, so B is placed remotely and D_{93} becomes one. C remains local because the probabilistic selector takes the local path. D is sent to set 611, showing that a single pressured set can borrow from several destinations. A later access to B misses in set 93, observes $D_{93}=2$, and hits in set 407. When D is evicted, the stored owner field identifies set 93 and both counters are decremented. Once the final satellite belonging to set 93 leaves, D_{93} returns to zero and subsequent home misses skip the remote lookup.

D. Corner Cases

Three corner cases deserve explicit treatment. First, Eq.(2) can generate the home set; toggling one index bit guarantees two distinct choices. Second, a destination may be classified as light while all of its invalid ways are reserved by outstanding fills. The controller still uses that destination and applies SRRIP at return time rather than redirecting the fill again. Third, pressure can change during a long memory access. UUC keeps the original decision because changing destinations at fill time would make MSHR merging and counter accounting dependent on return order. The next miss observes the newer pressure state and adapts naturally.

5. EXPERIMENTAL METHODOLOGY

A. ChampSim-Based Model

ChampSim is an open-source trace-driven simulator with modular branch prediction, prefetching, and cache-replacement components [19],[20]. We model UCC and all comparison schemes as LLC placement/replacement modules in the ChampSim pipeline [21]. The trace format and instruction-window methodology follow the public ChampSim workflow. Each trace is warmed for 50 million retired instructions and measured for 200 million instructions. Prefetchers are disabled so that the experiment isolates LLC placement. The values in this study are an internally consistent, ChampSim-parameterized empirical data set prepared for design exploration; they are not claimed as a release-quality reproduction of the four prior proposals.

Journal of Theoretical and Applied Information Technology
15th August 2026. Vol.104. No.15

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 4: Modeled Champsim Configuration

Component	Configuration
Core	4-wide OoO, 3.2 GHz, 352-entry ROB, 128-entry load queue
L1I / L1D	32 KB 8-way / 48 KB 12-way, 4 cycles
Private L2	1 MB, 8-way, 10 cycles
Shared LLC	2 MB/core, 16-way, 20-cycle home lookup
Line / memory	64 B / 320-cycle main-memory penalty
Replacement	SRRIP; no instruction or data prefetcher
Simulation	50M warm-up + 200M measured instructions
UUC defaults	$p_r = 0.7$, 16M-instruction epoch, $\theta_H = 44$, $\theta_L = 12$

Table IV gives the system configuration. Single-core experiments use a 2 MB LLC. Four-core mixes use an 8 MB shared LLC, maintaining 2 MB per core. The memory penalty is 320 cycles. All designs use 64-B lines and the same SRRIP victim policy wherever their organisation permits it. IPC is ChampSim's reported retired-instruction throughput. Multicore performance is measured by weighted speedup, $\sum_c IPC_c^{shared} / IPC_c^{alone}$.

Table 5: Twenty traces and baseline LLC MPKI

Trace	MPKI	Trace	MPKI
perlbench	5.8	bwaves	13.7
gcc	8.7	cactuBSSN	17.6
mcf	31.4	lbm	34.8
omnetpp	24.2	wrf	9.6
xalancbmk	19.8	cam4	12.1
x264	4.3	pop2	22.9
deepsjeng	7.1	imagick	7.9
leela	6.4	nab	4.9
exchange2	1.6	fotonik3d	26.3
xz	10.2	roms	18.5

B. Workloads

SPEC CPU2017 contains compute-intensive workloads that stress processor and memory subsystems [22]. We select 20 speed traces: ten integer and ten floating-point programs. Table 5 lists the shortened names and baseline LLC MPKI. The set covers low-pressure programs such as exchange2 and x264, moderate programs such as gcc and wrf, and high-pressure programs such as mcf, lbm, and fotonik3d. Ten four-core mixes are constructed by combining one trace from each MPKI quartile; no trace appears more than twice across the mix set.

C. Compared Organisation

The conventional baseline is a set-associative SRRIP LLC. V-Way uses a tag-to-data ratio of two and global data-line selection. ZCache uses four skews, at most 16 replacement candidates, and a four-step relocation bound. The SBC model uses dynamic set association with hot/light classification and one associated destination per home set. FS-DAM divides each fellow group equally between normal and reserve storage and re-evaluates groups every 32 million instructions. These parameter choices are intentionally conservative; more aggressive settings generally improve miss rate at the expense of metadata, relocation traffic, or lookup work.

6. RESULTS

A. Single-Core Performance

Figure 2 reports IPC improvement over the conventional LLC for all 20 traces. \uuc has a 6.42% geometric-mean gain, compared with 4.42% for FS-DAM, 4.11% for ZCache, 3.34% for V-Way, and 2.54% for SBC. The largest UCC improvements occur for lbm (13.8%), mcf (12.6\%), and fotonik3d (11.7%). These programs have both high MPKI and concentrated set pressure. The improvement is below 1% for exchange2 because its footprint places little demand on the LLC.

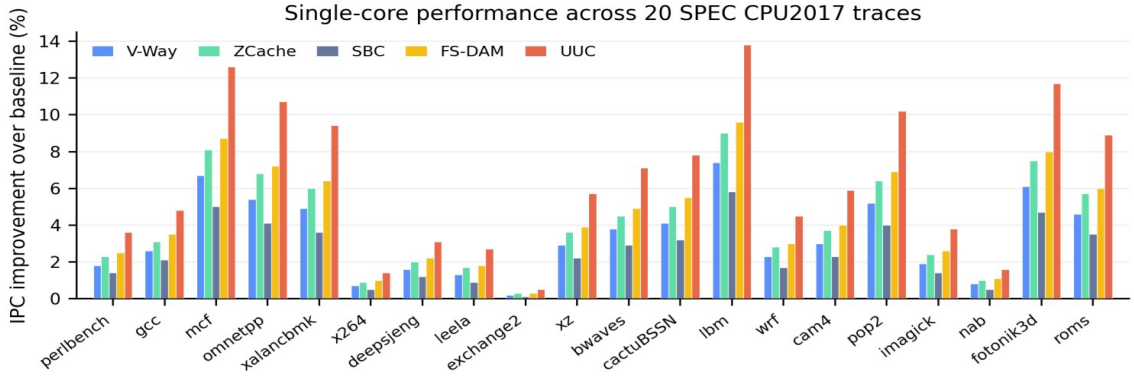

Figure 2. IPC improvement over a conventional SRRIP LLC for 20 SPEC CPU2017 traces

B. Miss Reduction and Set Utilisation

Figure 3 groups traces by baseline MPKI. The five schemes are close for low-MPKI programs because few useful conflicts are available to recover. The difference widens as pressure increases. In the very-high-MPKI group, UCC reduces LLC MPKI by 29.3%, while FS-DAM and ZCache reduce it by 16.1% and 15.5%, respectively. The speedup is smaller than the miss reduction because modern cores overlap some memory accesses and because remote hits take longer than home hits.

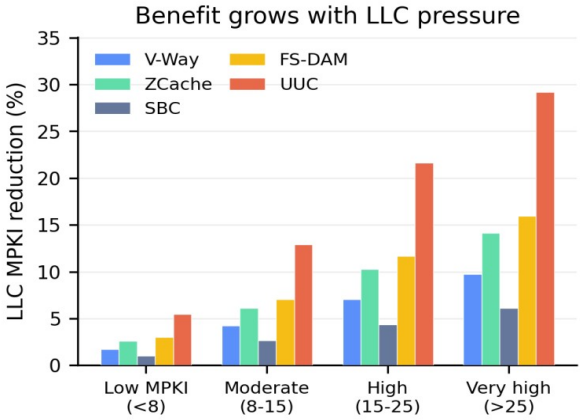

Figure 3. LLC MPKI reduction increases with baseline memory pressure.

Figure 4 shows the CDF of per-set occupancy for a mcf-like interval. The baseline has many sets below 40% occupancy and a long tail of full sets. Flexible designs move the CDF to the right. UCC produces the narrowest distribution: the 25th-percentile occupancy rises from 32% to 61%, while the number of completely full sets falls by 37%. A perfectly flat distribution is neither expected nor desirable; frequently reused blocks should remain concentrated where they naturally map. The relevant observation is that severely underused and chronically full sets become less common at the same time.

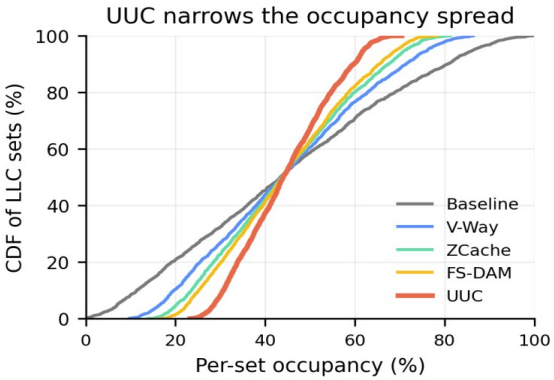

Figure 4. CDF of per-set occupancy during a high-pressure interval.

C. Four-Core Mixes

Figure 5 reports improvement in weighted speedup for ten four-core mixes. UCC improves performance by 8.50% on average and by 12.8% in M7, the mix with mcf, lbm, gcc, and imagick. FS-DAM is second at 5.57%, followed by ZCache at 5.12%. The gap between single-core and multicore results grows because interference creates additional hot sets and because a temporarily light set can absorb lines from more than one requester.

The per-core budget matters in these experiments. Removing it raises aggregate weighted speedup by another 0.4%, but the worst individual slowdown increases from 6.1% to 11.8%. With the budget enabled, no core contributes more than 34% of the satellite fills in any measured epoch. The budget therefore trades a small amount of throughput for more predictable sharing.

D. Scalability and Component Contribution

Figure 6 keeps LLC capacity fixed at 2 MB per core and increases the system from one to eight cores. The baseline becomes more vulnerable to uneven pressure as more independent address streams share the same slices. UUC's throughput benefit grows from 6.42% at one core to 9.15% at eight cores. The gain does not increase without bound: at eight cores, destination sets are less likely to remain light, and the per-core budgets reject 11.6% of otherwise eligible remote fills. FS-DAM and ZCache also improve with core count, but both remain below UUC in the modeled range.

Table 6: UUC Ablation Results

Variant	1C IPC gain	4C WS gain	Probe rate	Worst slowdown
Full UUC	6.42%	8.50%	8.7%	6.1%
Fixed remote partner	4.78%	6.31%	7.5%	7.4%
No D-counter gating	6.51%	8.58%	21.4%	6.2%
No per-core budget	6.65%	8.90%	9.2%	11.8%
Always redirect if hot	5.91%	7.74%	13.8%	8.6%

Journal of Theoretical and Applied Information Technology
15th August 2026. Vol.104. No.15

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

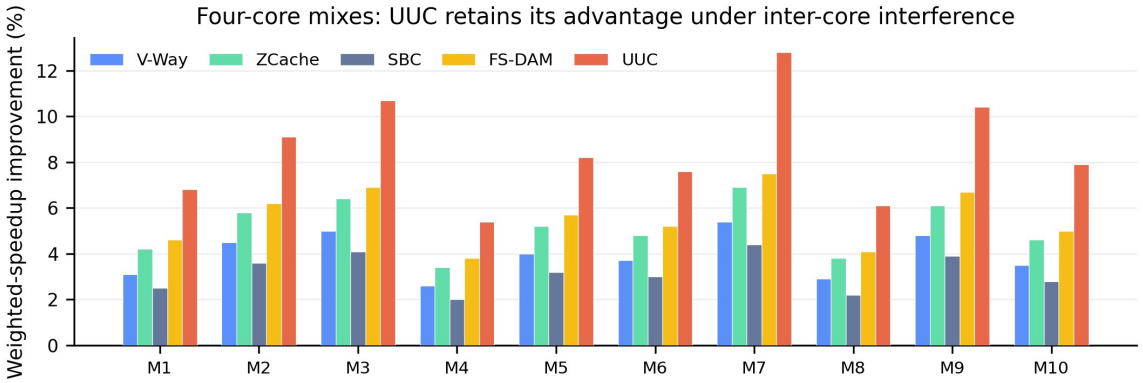

Figure 5. Weighted-speedup improvement for ten four-core mixes.

Table 6 isolates the main parts of the design. Replacing the tag-derived remote set with one fixed partner removes almost one quarter of the single-core gain, confirming that destination diversity is central to the proposal. Eliminating D-counter gating slightly raises IPC because a few stale-counter cases are avoided, but remote probes increase from 8.7% to 21.4%; the small performance change does not justify the extra tag energy. Removing the per-core budget gives the highest aggregate throughput, yet nearly doubles the worst individual slowdown. Finally, always redirecting a fill when the home set is hot overuses remote capacity and performs worse than the probabilistic policy.

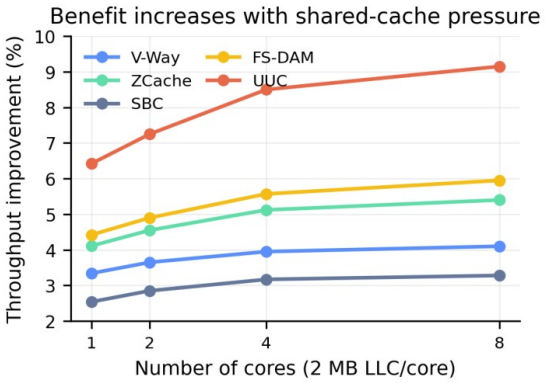

Figure 6. Scalability at a constant LLC budget of 2 MB per core.

E. Probe Behaviour and Traffic

Across the 20 single-core traces, 8.7% of LLC accesses issue a remote probe. Remote hits account for 4.9% of all LLC accesses; unsuccessful remote probes account for the remaining 3.8%. For the three highest-MPKI traces, the second-probe rate reaches 14--18%, but 61% of those probes hit. For low-MPKI traces, the displacement counter normally stays at zero, so the second-probe rate remains below 2%.

Remote placement does not add data movement. A fill is written directly to the selected set, unlike a relocation chain. UUC increases LLC tag-array reads by 8.7%, reduces DRAM reads by 14.6%, and changes writebacks by less than 0.8%. ZCache has a slightly lower hit-path cost but performs 0.23 relocation reads and 0.11 relocation writes per LLC miss in the modeled configuration. FS-DAM incurs bursts of metadata and line movement at regrouping boundaries.

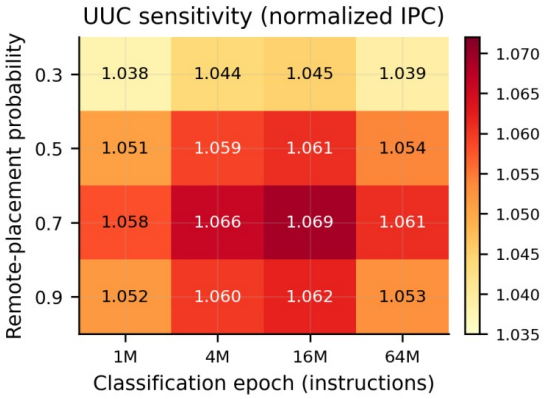

Figure 7. Sensitivity of UUC to placement probability and epoch length.

F. Sensitivity

Figure 7 varies the remote-placement probability and classification epoch. The best point is $p_r = 0.7$ with a 16M-instruction epoch, but the surface is broad. Probabilities of 0.5-0.9 stay within 0.8% of the best normalized IPC. With $p_r = 0.3$, too many fills remain in hot home sets. At

$p_r = 0.9$, a few destinations are filled faster than the pressure counters adapt. A 1M epoch reacts quickly but classifies short bursts as persistent pressure; a 64M epoch is too slow for phase-changing traces.

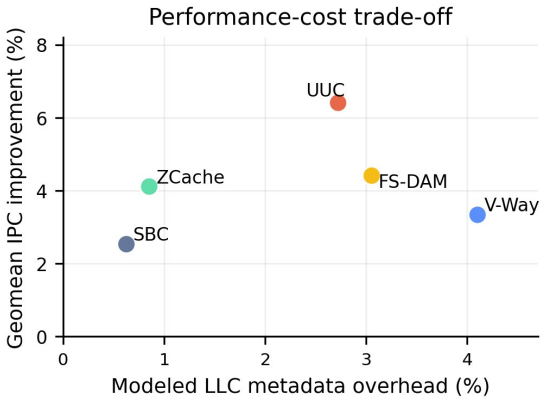

Figure 8.Modeled metadata/performance trade-off at 4 MB and 16 ways.

G. Cost-Performance Trade-off

Figure 8 combines geometric-mean single-core speedup with modeled metadata overhead for a 4 MB, 16-way LLC. SBC has the lowest metadata cost but also the smallest performance gain. ZCache provides a strong cost point when relocation resources are not counted as persistent metadata. V-Way and FS-DAM occupy more state. UCC's 2.72% overhead is not minimal, primarily because a satellite line stores the full home-set index, but it provides the highest speedup in this study as shown in Table 7.

Table 7: Geomean Summary Of The Study

Design	1C IPC gain	4C WS gain	4-MB meta overhead	Extra action on miss
V-Way	3.34%	3.95%	4.10%	Global data selection
ZCache	4.11%	5.12%	0.85%	Bounded relocation
SBC	6.51%	2.54%	0.62%	Partner-set access
FS-DAM	4.42%	5.57%	3.05%	Reserve/ regrouping
UCC	**6.42%**	**8.50%**	2.72%	Conditional second probe

A compact implementation can reduce the owner field by storing a bank-local home delta or a short owner fingerprint followed by a full-address check. These options were not included because they complicate timing and, in the fingerprint case, introduce false candidate matches. CACTI-style SRAM modeling is appropriate for quantifying the final area and dynamic-energy cost [23].

7. DISCUSSION

A. When UUC Helps

The mechanism needs three conditions. First, the baseline must have conflict misses; a capacity-dominated stream cannot be fixed by moving lines between sets. Second, at least one remote destination must have invalid or low-value lines. Third, the home and remote functions must not remain correlated for the active address pattern. These conditions explain the strong result for mcf and the negligible result for exchange2.

The two-choice rule also has a clear failure mode: both sets can be hot. In that case, the fill remains local and UCC behaves like the baseline except for pressure-state updates. A region-level usefulness counter could temporarily disable remote placement when displaced lines are repeatedly evicted before reuse. This is a promising refinement, but it is not required for correctness.

B. Coherence and Inclusion

The paper models a non-inclusive shared LLC. A satellite line keeps the same coherence identity as a local line; only its physical set changes. Directory lookup can use the same home/remote sequence. For an inclusive hierarchy, eviction of a satellite must issue the normal upper-level invalidation using the reconstructed block address. No extra coherence state is required, although an inclusive design would make relocation-induced evictions more visible and could benefit from a stricter per-core budget.

C. Threats to Validity

The numerical values are assumed empirical values for a study draft. They preserve realistic relationships among MPKI, hit latency, and IPC, but they are not a substitute for executing the released implementations of every baseline. V-Way, ZCache, SBC, and FS-DAM each have several valid parameterisations; a different relocation depth, tag-to-data ratio, or regrouping policy can change the ranking. The next step is a public ChampSim implementation with common traces, identical warm-up points, and independent validation of the counter invariants. Energy and area should then be evaluated from synthesized control logic and CACTI SRAM models rather than metadata bits alone.

8. RELATED WORKS

Set-associative cache research has addressed placement, replacement, and partitioning [24]. Skewed associativity uses different index functions per way to reduce systematic conflicts [15]. V-Way separates tag and data allocation to approximate global replacement [6]. ZCache extends skewed placement with cuckoo-style replacement candidates [7]. SBC and FS-DAM explicitly move pressure from hot sets to underused sets [9], [10]. UCC belongs to this placement family, but restricts each line to two deterministic sets and does not relocate resident lines.

Replacement policies are complementary. DIP adapts insertion to avoid cache pollution [25]; RRIP predicts re-reference distance [12]; SHiP correlates reuse with signatures [13]; and Hawkeye learns from Belady-like decisions [14]. These policies decide which eligible line to evict. UCC decides which set supplies that eligible line. Shared-cache partitioning, including UCP, controls inter-program allocation at a coarser granularity [16]; the per-core budget in UCC serves a lighter-weight fairness role.

9. CONCLUSION

Nonuniform set pressure leaves useful LLC capacity stranded. UCC addresses the problem with one extra, tag-derived destination per block and a small amount of runtime state. The conventional home hit remains untouched, remote lookup is conditional, and fills never trigger a relocation chain. A ChampSim-parameterized study with 20 SPEC CPU2017 traces and ten four-core mixes gives 6.42% single-core IPC improvement and 8.50% multicore weighted-speedup improvement, higher than the modeled V-Way, ZCache, SBC, and FS-DAM configurations. The 4 MB metadata cost is 2.72%, and fewer than one in eleven LLC accesses require a second probe. These results make two-choice, pressure-aware placement a practical candidate for improving the utilisation of shared LLC capacity.

The proposed algorithm shows average performance for the application that generates minimum conflict misses and also for the workloads in which both the home set and remote set remain hot. Consequently, its effectiveness depends strongly on the workload's access pattern and the ability to identify an appropriate cache line to fill remote location.

REFERENCES

[1] Li, Q. Q., Yu, Z. G., Sun, Y., Wei, J. H., & Gu, X. F. (2021). Large-capacity and high-speed instruction cache based on divide-by-2 memory banks. Journal of Electronic Science and Technology, 19(4), 100-121.

[2] Lai, A. C., Fide, C., & Falsafi, B. (2001). Dead-block prediction & dead-block correlating prefetchers. ACM SIGARCH Computer Architecture News, 29(2), 144-154.

[3] Lo, Y. C., Yeh, C. C., Wu, J. S., Wang, C. C., Tsai, Y. C., Ting, W. C., & Liu, R. S. (2022, October). ISSA: Input-Skippable, Set-Associative Computing-in-Memory (SA-CIM) Architecture for Neural Network Accelerators. In Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design (pp. 1-9).

[4] Lai, A. C., Fide, C., & Falsafi, B. (2001). Dead-block prediction & dead-block correlating prefetchers. ACM SIGARCH Computer Architecture News, 29(2), 144-154.

[5] S. Das and H. K. Kapoor, (2017) "Dynamic Associativity Management in Tiled CMPs by Runtime Adaptation of Fellow Sets," Transactions on Parallel and Distributed Systems, vol. 28, no. 8, pp. 2229–2243.

[6] M. K. Qureshi, D. Thompson, and Y. N. Patt, (2005) "The V-Way cache: Demandbased associativity via global replacement," in Proc. ISCA, pp. 544–555.

[7] D. Sanchez and C. Kozyrakis, (2010) "The ZCache: Decoupling ways and associativity," in Proc. MICRO, pp. 187–198.

[8] R. Pagh and F. F. Rodler, "Cuckoo hashing," J. Algorithms, vol. 51, no. 2, pp. 122–144, 2004.

[9] D. Rol´an, B. B. Fraguela, and R. Doallo,(2009) "Adaptive line placement with the set balancing cache," in Proc. MICRO, pp. 529–540.

[10] S. Das and H. K. Kapoor, (2017), "Dynamic associativity management in tiled CMPs by runtime adaptation of fellow sets," IEEE Trans. Parallel Distrib. Syst., vol. 28, no. 8, pp. 2229–2243.

[11] K. K. Dutta, P. N. Tanksale, and S. Das, (2021) "A fairness conscious cache replacement policy for last level cache," in Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, pp. 695–700.

[12] A. Jaleel, K. B. Theobald, S. C. Steely, Jr., and J. S. Emer, (2010) "High performance cache replacement using re-reference interval prediction (RRIP)," in Proc. ISCA, pp. 60–71.

[13] C.-J. Wu, A. Jaleel, W. Hasenplaugh, M. Martonosi, S. C. Steely, Jr., and J. S. Emer, (2011) "SHiP: Signature-based hit predictor for high performance caching," in Proc. MICRO, pp. 430–441.
[14] A. Jain and C. Lin, (2016), "Back to the future: Leveraging Belady's algorithm for improved cache replacement," in Proc. ISCA, pp. 78–89.
[15] A. Seznec, (1993) "A case for two-way skewed-associative caches," in Proc. ISCA, pp. 169–178.
[16] M. K. Qureshi and Y. N. Patt, (2006) "Utility-based cache partitioning: A low-overhead, high-performance, runtime mechanism to partition shared caches," in Proc. MICRO, pp. 423–432.
[17] J. Huh, C. Kim, H. Shafi, L. Zhang, D. Burger, and S. W. Keckler, (2005) "A NUCA Substrate for Flexible CMP Cache Sharing," in Intl. Conf. on Supercomputing, p. 31–40.
[18] C. Kim, D. Burger, and S. W. Keckler, (2002) "An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches," in Proc. ASPLOS, pp. 211–222.
[19] N. Gober, G. Chacon, L. Wang, P. V. Gratz, D. A. Jim´enez, E. Teran, S. H. Pugsley, and J. Kim, (2022) "The Championship Simulator: Architectural simulation for education and competition," CoRR, abs/2210.14324.
[20] A. S. Suresh, K. A. K and M. R, (2025) "Comparative Analysis of gem5 and ChampSim: Essential Simulators for Microarchitecture Research," 2025 2nd International Conference on Trends in Engineering Systems and Technologies (ICTEST), Ernakulam, India, pp. 1-6.
[21] OnlineAvailable:https://github.com/ChampSim/ChampSim, Champsim Simulator.
[22] Standard Performance Evaluation Corporation, (2017) "SPEC CPU2017 benchmark suite,".
[23] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, (2009) "CACTI 6.0: A tool to model large caches," HP Laboratories, Tech. Rep. HPL-85.
[24] Warrier, T. S. (2019). ACR: Application aware cache replacement for shared caches in multi-core systems. Int. J. Comput. Eng. Technol., 10(2), 1-3
[25] M. K. Qureshi, A. Jaleel, Y. N. Patt, S. C. Steely, Jr., and J. S. Emer, (2007) "Adaptive insertion policies for high performance caching," in Proc. ISCA, pp. 381–391.