Computational scaling of steady-state thermal network solves in data center buildings from 8 to 1,024 racks

Ricardo Marquez, Ph.D.
rmarquez123@gmail.com

September 2026

Abstract

Thermal models of data center buildings are only useful at the level of the individual rack if they can still be solved when a building holds a thousand racks. This report measures how the cost of a steady-state solve of such a model grows with building size. It is a forward study: every input is stated and nothing is estimated from data. A building of four rooms is scaled from 8 to 1,024 racks (34 to 3,082 unknowns), with every heat path sized at every size for the heat it carries, and the resulting linear network is solved by a full sparse Newton solve and by a solve that eliminates each rack's unknowns and then solves the remaining interface system. Both are timed against a dense LU solve of a random matrix of the same size. The two structured solves agree to better than 10^{-11} relative on every unknown at every size, and each converges in one Newton step. Counted in floating-point operations, eliminating the rack blocks saves a factor that rises from 7.90 to 26.56, toward a limit of 27. Measured on one laptop, the full sparse solve grows as $N^{0.82}$ and the dense reference as $N^{2.40}$ over the eight sizes, and between the two largest sizes the local slopes are 1.21 and 2.91. At 1,024 racks the full sparse solve is at least 20× (20–23× over five repeats) faster than the dense reference (27.3 ms against 558 ms in the plotted repeat); the dense reference is faster up to 128 racks. The elimination solve gives the same answer but is slower than the full sparse solve at every size.

1 Introduction

Data centers are a visible and growing share of electricity demand: Masanet, Shehabi, Lei, Smith and Koomey (2020) estimate them at about one percent of global electricity use, part of it spent removing heat. Operating and designing that cooling well needs thermal models, and the decisions that matter most — which rack runs hot, where a new load can go, how close a room is to its limit — are made rack by rack. A building-scale model that resolves every rack has thousands of coupled unknowns, and a general-purpose dense solver pays for them with a cost that grows as the cube of their number: doubling the building multiplies the work by eight. The usual way around that cost is to lump racks into zone averages, which hides the very rack a decision is about.

The alternative is old and well understood in numerical linear algebra. Parter (1961) described Gaussian elimination as an operation on the graph of the matrix and showed that the order of elimination decides how much fill it creates; on a tree, eliminating leaves first creates none. George (1973) showed that ordering the elimination of a regular two-dimensional mesh by recursive separators reduces its factorization to order $N^{3/2}$ operations, and Davis (2006) gives the theory and algorithms of sparse direct solvers built on such orderings; the supernodal partial-pivoting method of Demmel, Eisenstat, Gilbert, Li and Liu (1999) is the sparse factorization used here. Eliminating independent blocks and solving the system left on their shared interface is the Schur-complement

construction that underlies domain decomposition (Saad 2003). Dense LU factorization, the reference here, is the LAPACK routine (Anderson et al. 1999). On the building side, whole-building simulation programs such as EnergyPlus (Crawley et al. 2001) and equation-based libraries such as the Modelica Buildings library (Wetter et al. 2014) solve coupled thermal networks of buildings, and data-center studies have used reduced linear thermal models when speed mattered, for example the heat-recirculation model behind the thermal-aware scheduling of Tang, Gupta and Varsamopoulos (2008). None of the numerical constructions in this report is new, and none is claimed as new.

TERRA is a general-purpose engine for probabilistic inference on physical networks. It treats a network as a graph on which the physics is written locally: a conservation law at every node, a constitutive law on every edge, and the boundary and component laws that close them, attached to the same graph. Each of these is a relation the state of the network must satisfy, and the engine assembles them and solves them together. Domains enter through plugins that supply their laws, units and file formats; the solving and the bookkeeping are shared. This study uses its data-center plugin to compose a building network from rack, room and building components, solves that network with the engine's sparse Newton solver and with its interface-elimination mode, and counts and times the cost of both as the building grows. Section 3 states the framework and each of these constructs precisely.

This report asks one question: how does the cost of solving a rack-resolved building network grow with the number of racks when the solve follows the network's tree structure, against a dense factorization of a system of the same size? The construction is a sweep over eight building sizes, solved two structured ways, checked for agreement and for physical validity, counted in operations and timed. It is a forward study: nothing is fitted, and the only measured quantities are wall-clock times. What it adds is a scaling measurement on a network whose every link is sized for the heat it carries, so that each solved building is one a real building could run at, with the dense reference stated exactly. Section 2 describes the building and its model, Section 3 the relations, the two solves, the timing protocol and the measures, Section 4 the results, Section 5 their limits and Section 6 the answer; the appendices hold the reproduction recipe and the provenance of the numbers.

2 System and data

The building. Figure 1 draws the system. A building of four rooms rejects its heat through a single building air node to chilled water held at 290 K. Every room holds the same number K of racks, and K is swept through 2, 4, 8, 16, 32, 64, 128, 256, so the building grows from 8 to 1,024 racks while its shape stays the same. Each rack draws a base load that rises by 0.5 W from one rack to the next, starting at 30.0 W, so that no two racks are identical, plus a small extra load that grows with the rack's temperature. Table 1 lists every input.

Data. No measured data enter this report. The building and its loads are constructed, and every quantity except the solve times is a deterministic function of the inputs of Table 1. The solve times were measured on one development laptop under the protocol of Section 3.6; Appendix B records the machine.

The model. Figure 2 shows the unknowns and relations along one path from a rack to the chilled water. Each rack has a temperature, a load and a heat flow to its room; each room an air temperature and a heat flow to the building; the building an air temperature and a heat flow to

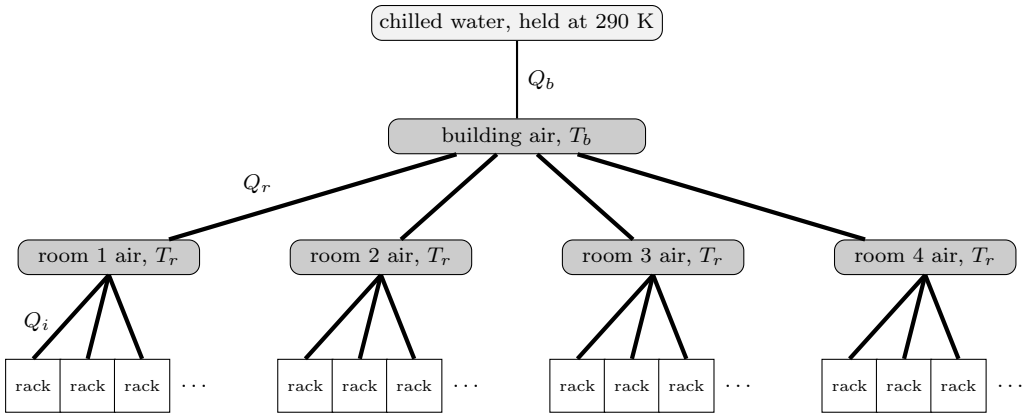


Figure 1: The building of this report. A single building air node rejects heat to chilled water held at 290 K; four rooms reject heat to the building air; and every room holds the same number K of racks, each rejecting its heat to its own room's air. A rack is connected to nothing but its room, and a room to nothing but its racks and the building. The heavy links and shaded boxes are the quantities through which the racks and rooms couple to the rest of the building — the rack heat flows Q_i, the room air temperatures T_r and heat flows Q_r, and the building air temperature T_b. Whatever the number of racks, that is all a rack shares with the rest of the building.

the chilled water. The relations are a component law for each rack's load, a conservation law at each node and a constitutive law on each link, closed by the given chilled-water temperature.

Assumptions. The model rests on the following choices.

- Steady state: no thermal storage anywhere in the building.
- Linear physics: a rack's extra load is linear in its temperature with slope 0.1 W/K, and every link carries heat in proportion to the temperature difference across it.
- A flat layout: racks connect directly to their room's air, with no rows or aisles in between.
- The chilled-water temperature is held fixed; the chiller plant is not modeled.
- Each link's conductance is sized at every building size from that size's heaviest rack (Section 3.3).
- Units are kelvin, watts and watts per kelvin throughout.

3 Methods

3.1 The network model

The building network of Section 2 is one instance of a general construction, stated first. A physical network is a graph, and the engine writes its physics on that graph. The state is a vector z of unknowns: one potential per node, such as a temperature or a head, one flow per edge, and whatever else the physics determines. Every relation the state must satisfy — a conservation law

Table 1: Inputs of the building network. The conductances are sized at every building size from that size's design rack (Eqs. (11)–(14)); the values shown are those of the largest building.

Quantity	Symbol	Value
Rooms	R	4
Racks per room	K	2, 4, 8, 16, 32, 64, 128, 256
Lightest rack's base load	P_0	30.0 W
Base-load step between racks	ΔP	0.5 W
Leakage slope	α	0.1 W/K
Reference temperature of the base load	T_{ref}	300.0 K
Chilled-water temperature	T_{chw}	290.0 K
Rack air-temperature rise at design	ΔT_{rise}	12.0 K
Room and building approaches at design	ΔT_{room}, ΔT_{bldg}	3.0 K, 3.0 K
Design rack load at 1,024 racks	P_d	541.5 W
Link conductances at 1,024 racks	G_{rk}, G_{room}, G_{bldg}	45.125; 46,208; 184,832 W/K

at a node, a constitutive law on an edge, a boundary condition, a component law — is written as a residual $r_k(z)$ that vanishes when the relation holds. With as many independent relations as unknowns and no readings to reconcile, the state is the unique solution of

$$r(z) = 0, \tag{1}$$

found by Newton iteration on the sparse Jacobian J of the residuals, $J_{ki} = \partial r_k / \partial z_i$, from a stated start until the residual norm falls below a tolerance ϵ:

$$z \leftarrow z - J^{-1} r(z) \quad \text{until } \|r(z)\| < \epsilon. \tag{2}$$

A transient model adds a storage term to the conservation relations and advances the state in time with an implicit integrator, solving a system of the same kind at every step. Nothing in this report is estimated from data: every input is stated, and the results are the solutions of these relations and quantities computed from them. Everything that follows is a choice of unknowns and relations on this graph.

3.2 Relations of the building network

Index the rooms by $r = 0, \ldots, R-1$ with $R = 4$ and the racks of a room by $k = 0, \ldots, K-1$, and write i for a rack, $n = RK$ for the number of racks, and $P_0 = 30.0$ W and $\Delta P = 0.5$ W for the lightest base load and its step. Rack k of room r has the base load

$$P_{\mathrm{base},i} = P_0 + \Delta P\,(rK + k). \tag{3}$$

Its load rises with its temperature T_i about $T_{\mathrm{ref}} = 300$ K with slope $\alpha = 0.1$ W/K; the whole load leaves as the heat flow Q_i to its room; the room passes the sum of its racks' heat to the building; and the building passes the sum of its rooms' heat to the chilled water. Each link carries heat in

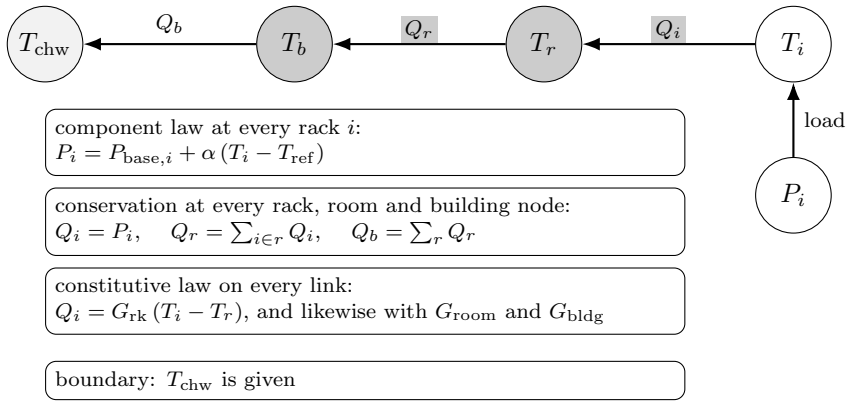


Figure 2: The unknowns and the relations that tie them, along one path from a rack to the chilled water. Circles are temperatures and the rack's load; labels on the arrows are the heat flows, which are unknowns as well. White marks the unknowns interior to a rack, shaded gray the interface unknowns of Figure 1, and the lightly shaded circle the given chilled-water temperature; the building's heat flow Q_b is interior too. Every rack contributes three unknowns and three relations (its load law, its balance and its link), every room two, and the building two, so the system is square at every size.

proportion to the temperature difference across it:

$$P_i = P_{\text{base},i} + \alpha\,(T_i - T_{\text{ref}}), \tag{4}$$
$$Q_i = P_i, \tag{5}$$
$$Q_i = G_{\text{rk}}\,(T_i - T_r), \tag{6}$$
$$Q_r = \textstyle\sum_{i \in r} Q_i, \tag{7}$$
$$Q_r = G_{\text{room}}\,(T_r - T_b), \tag{8}$$
$$Q_b = \textstyle\sum_r Q_r, \tag{9}$$
$$Q_b = G_{\text{bldg}}\,(T_b - T_{\text{chw}}). \tag{10}$$

with $T_{\text{chw}} = 290$ K given. Equations (4)–(6) hold at every rack, (7) and (8) at every room, and (9) and (10) once; they are the residuals $r(z)$ of Eq. (1). Every relation is linear in the unknowns, so the Newton iteration (2) reaches the solution in a single step from any start. Both solves start from 305 K at every rack, 320 K at every room and 310 K at the building, and stop when the residual norm falls below 10^{-8} or below 10^{-9} of its initial value.

3.3 Sizing of the heat paths

The conductances are not fixed numbers but are sized at every building size for the heat each link carries at design. The design rack is the building's heaviest, and a rack's air stream is sized for a 12 K temperature rise at that load; since an air stream of mass flow $\dot{m}$ and specific heat c_p has conductance $\dot{m}c_p$ and carries $P_d = \dot{m}c_p\Delta T_{\text{rise}}$, its conductance is $P_d/\Delta T_{\text{rise}}$. A room's link carries the heat of its K design racks at a 3 K approach, and the building's link all RK design racks at a

3 K approach:

$$P_d = P_0 + \Delta P\,(RK - 1), \tag{11}$$
$$G_{\mathrm{rk}} = P_d/\Delta T_{\mathrm{rise}}, \tag{12}$$
$$G_{\mathrm{room}} = KP_d/\Delta T_{\mathrm{room}}, \tag{13}$$
$$G_{\mathrm{bldg}} = RKP_d/\Delta T_{\mathrm{bldg}}. \tag{14}$$

At 1,024 racks this gives $P_d = 541.5$ W and conductances of 45.125, 46,208 and 184,832 W/K. Sizing changes the coefficients of Eqs. (4)–(10) but never which unknowns a relation contains, so it leaves the structure measured below unchanged.

3.4 Structure of the system

Each rack contributes three unknowns (T_i, P_i, Q_i), each room two (T_r, Q_r) and the building two (T_b, Q_b). The elimination solve keeps as its interface the quantities through which a rack or a room reaches the rest of the building: every rack heat flow, every room's temperature and heat flow, and the building temperature. The remaining unknowns are interior, and they fall into independent blocks: one block of two unknowns (T_i, P_i) per rack, which no relation shares with another rack, and one block holding the building's heat flow Q_b. With N the number of unknowns, m the interface size and b the number of interior blocks,

$$N = 3n + 2R + 2, \tag{15}$$
$$m = n + 2R + 1, \tag{16}$$
$$b = n + 1. \tag{17}$$

so the interior holds $2n + 1$ unknowns. Figure 3 draws the Jacobian with the interior ordered first: a block-diagonal interior bordered by its couplings to the interface.

3.5 Full sparse and interface-elimination solves

The *full sparse solve* takes each Newton step of Eq. (2) with a sparse LU factorization of the whole Jacobian (the supernodal method of Demmel et al. 1999, through SciPy; Virtanen et al. 2020), whose fill-reducing column ordering is free to follow the tree. The *interface-elimination solve* partitions the unknowns into the interior I and the interface B of Section 3.4, pairing each unknown with a relation by a structural matching so that J_{II} is nonsingular, and computes each Newton step as

$$S = J_{BB} - J_{BI}J_{II}^{-1}J_{IB}, \tag{18}$$
$$S\,\Delta z_B = -\big(r_B - J_{BI}J_{II}^{-1}r_I\big), \tag{19}$$
$$\Delta z_I = -J_{II}^{-1}\big(r_I + J_{IB}\,\Delta z_B\big), \tag{20}$$

where r_I and r_B are the residuals paired with the interior and the interface. J_{II}^{-1} is applied block by block, each interior block factorized on its own by the same sparse LU, and the interface system (19) is formed as a dense $m \times m$ matrix and solved by dense LU. The two solves compute the same state by different sequences of floating-point operations, so their difference measures round-off only.

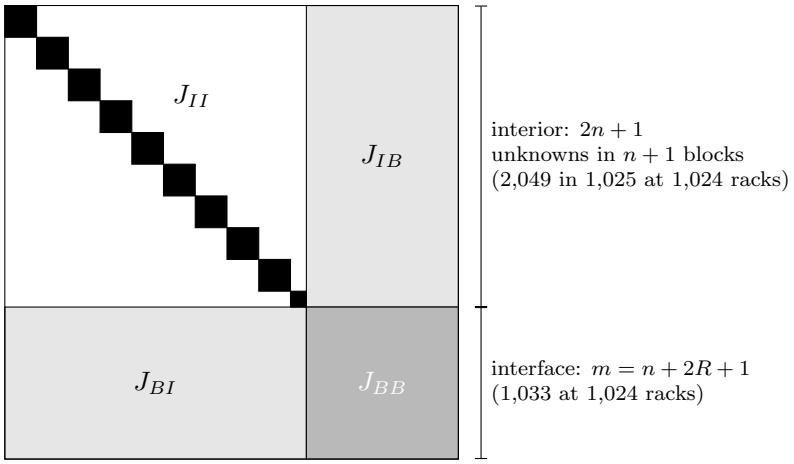


Figure 3: The Jacobian of the building network with the interior unknowns ordered first, drawn to the proportions of the largest building (1,024 racks, 3,082 unknowns). The interior block J_{II} is block-diagonal: each black square is one rack's temperature and load, which no relation shares with another rack, and the last, smaller square is the building's heat flow to the chilled water; only nine of the 1,025 blocks are drawn. The gray borders J_{IB} and J_{BI} couple each block to the interface, and J_{BB} couples the interface unknowns with each other. Eliminating the blocks one at a time leaves a system in the interface unknowns alone.

3.6 Dense reference and timing protocol

The *dense reference* is the cost any solver pays that treats the system as one undifferentiated block. At each size it is a call to NumPy's dense linear solver (Harris et al. 2020), an LU factorization with partial pivoting from LAPACK (Anderson et al. 1999), on a random dense symmetric positive-definite matrix of the same size N: $A = GG^\top + NI$, with G an $N \times N$ matrix of standard normal draws from a generator seeded with 0. It is not the network's own matrix; the cost of dense LU depends on the size of the matrix and not on its entries. Only the solve is timed, not the construction of A. The two structured solves are timed as complete solves from the compiled model: assembling the residual and the Jacobian, factorizing, stepping and checking convergence. Each timed value is the fastest of five solves. The whole sweep over the eight sizes was repeated five times on one development laptop with single-threaded BLAS, each repeat started only once no other Python, Cargo or LaTeX process was running and the whole machine's processor load, averaged over ten seconds, was below 30 %.

3.7 Metrics

Agreement. Whether the two structured solves compute the same state is measured by their largest relative difference over all unknowns,

$$\delta = \max_j \frac{|z_j^{\text{sparse}} - z_j^{\text{elim}}|}{\max(1, |z_j|)}, \tag{21}$$

dimensionless. It is a round-off quantity, so it is reported as a bound, not as a value; the acceptance limit is 10^{-10}, well above the round-off of systems of this size and far below any physically meaningful difference.

Physical validity. Every solved building is checked against the limits of Table 2: no rack above 320 K, no room air below 291.15 K, no rack's temperature-dependent load above 10 % of its base load, and the heat reaching the chilled water equal to the heat the racks produce, and every node's flows balanced, to within 10^{-9} relative. These limits separate a building that could operate from a network whose steady state has no physical meaning.

Operation count. The cost of the two approaches in floating-point operations, independent of any machine, is modeled as the cube of each dense system factorized: N^3 for the dense reference, and for the elimination the b interior blocks of average size $\bar{s}$ plus the dense interface system. The ratio

$$\bar{s} = (N - m)/b, \tag{22}$$

$$\rho = N^3/\left(b\,\bar{s}^3 + m^3\right) \tag{23}$$

is the factor the elimination saves. By Eqs. (15)–(17), N/m tends to 3 as the racks per room grow and the block term becomes negligible, so ρ tends to 27, the reference value to read it against.

Growth exponent. How the measured time t grows with size is summarized by the exponent p of the least-squares line

$$\log t = p \log N + c \tag{24}$$

through all eight sizes of one repeat, per solve. The reference values are $p = 3$ for dense LU of large matrices and $p = 1$ for a solve whose cost follows a tree. Because small solves are dominated by fixed overhead, the slope between the two largest sizes alone is also given, as a descriptive local exponent.

Speed bound. The speed of the full sparse solve relative to the dense reference at the largest size is the ratio $t_{\text{dense}}/t_{\text{sparse}}$. Wall-clock time varies between runs, so the published figure is a bound: the integer part of the ratio in each repeat, and its minimum over the five repeats,

$$B = \min_k \left\lfloor t^{(k)}_{\text{dense}}/t^{(k)}_{\text{sparse}} \right\rfloor, \tag{25}$$

with the range over the repeats printed beside it. The figures and tables of timings show the repeat that gave the minimum.

4 Results

4.1 Agreement of the two structured solves

At every one of the eight sizes the full sparse solve and the interface-elimination solve agree to $\delta < 10^{-11}$ on every unknown (Eq. (21)), an order of magnitude inside the acceptance limit of 10^{-10}, and both converge in one Newton step, as linear relations must (Table 3, last column). The two ways of organizing the elimination are interchangeable for the answer; the rest of this section is about their cost.

Table 2: Physical checks of the solved network at each building size. The leakage share is a rack's temperature-dependent extra load as a percentage of its base load; the last row gives the limits each column is checked against.

Racks n	Hottest rack [K]	Coldest rack [K]	Coldest room air [K]	Largest leakage share [%]
8	308.26	306.66	295.68	2.46
16	308.01	305.01	295.25	2.14
32	307.66	302.62	294.64	1.68
64	307.24	299.75	293.90	1.18
128	306.85	297.00	293.19	1.00
256	306.55	294.90	292.65	1.70
512	306.36	293.53	292.30	2.16
1,024	306.25	292.75	292.10	2.42
limit	≤ 320.00	—	≥ 291.15	≤ 10.00

4.2 Physical validity of the solved networks

Table 2 lists, at each size, the hottest and coldest rack, the coldest room air and the largest leakage share, with the limit each is checked against in the last row. Over the 2,040 rack temperatures of the sweep the hottest rack is at 308.26 K, below the 320 K limit. The hottest rack of every building lies between 306.25 and 308.26 K, close to the chilled water plus the 12 K rise and the two 3 K approaches its links are sized for, which is what sizing every building for its own design rack is meant to achieve. The coldest rack is at 292.75 K and the coldest room air at 292.10 K, above its 291.15 K limit. No rack's temperature-dependent load exceeds 2.5 % of its base load, against a limit of 10 %, and the heat and flow balances hold to within 10^{-9} relative at every size. Every solved building is one that could operate, so the costs below are those of physically meaningful solves.

4.3 Structure and operation counts

Table 3 lists, at each size, the number of unknowns, the interior blocks, the interface size, the interface as a share of the unknowns and the operation-count ratio ρ. The counts follow Eqs. (15)–(17) exactly, from 34 unknowns in 9 blocks with an interface of 17 at 8 racks to 3,082 unknowns in 1,025 blocks with an interface of 1,033 at 1,024. Because every rack's heat flow belongs to the interface, the interface does not stay small: its share of the unknowns falls from 50.0 % to 33.5 % and tends to one third. Figure 4 plots ρ against the number of racks. The ratio rises monotonically from 7.90 to 26.56 and flattens toward its limit of 27: counted this way, with the interface system treated as dense, eliminating the rack blocks buys a large constant factor over a dense factorization, not a factor that grows with the building. A widening gap requires a solver that also keeps the interface system sparse.

4.4 Measured solve times

Figure 5 plots the measured solve time of the dense reference, the full sparse solve and the interface-elimination solve against the number of unknowns, with the fitted lines of Eq. (24), and Table 4 lists the same times with the speed ratio of the full sparse solve over the dense reference. At the smallest sizes the dense reference is by far the fastest: at 8 racks it takes 0.0150 ms against 0.944 ms for the full sparse solve, whose time there is dominated by the fixed overhead of assembly and bookkeeping, and even falls from 0.944 to 0.644 ms between the two smallest sizes. The dense

Table 3: Structure of the system at each building size. Blocks are the independent interior blocks, interface the unknowns the elimination solve keeps, ρ the operation-count ratio of Eq. (23), and the last column the Newton steps of the full sparse and the interface-elimination solve. Every entry is exact and does not depend on the machine.

Racks n	Unknowns N	Blocks b	Interface m	m/N	ρ	Newton steps
8	34	9	17	50.0 %	7.90	1 / 1
16	58	17	25	43.1 %	12.39	1 / 1
32	106	33	41	38.7 %	17.22	1 / 1
64	202	65	73	36.1 %	21.16	1 / 1
128	394	129	137	34.8 %	23.78	1 / 1
256	778	257	265	34.1 %	25.30	1 / 1
512	1,546	513	521	33.7 %	26.13	1 / 1
1,024	3,082	1,025	1,033	33.5 %	26.56	1 / 1

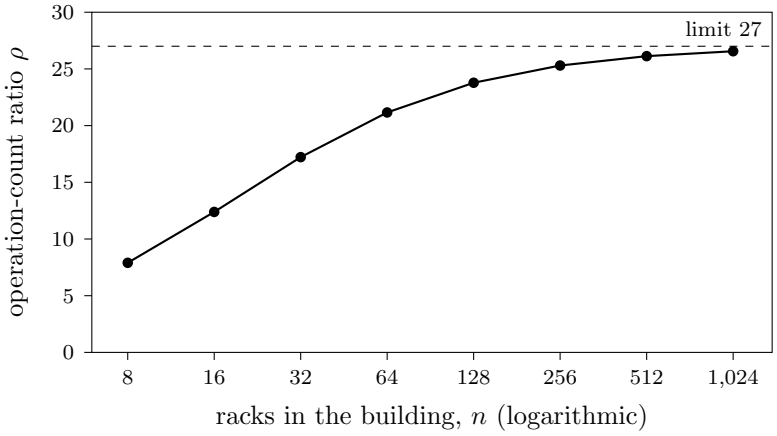


Figure 4: The operation-count ratio ρ of Eq. (23) at each of the eight building sizes, with the horizontal axis logarithmic in the number of racks. Markers are the ratios; the dashed line is their limit of 27 as the racks per room grow. The ratio rises from 7.90 at 8 racks to 26.56 at 1,024 and flattens toward the limit: eliminating the blocks saves a large constant factor over a dense factorization, not a factor that keeps growing.

reference remains faster up to 128 racks (2.01 against 2.83 ms) and is slower from 256 racks on (12.2 against 6.46 ms). Over the eight sizes the fitted exponents are 2.40 for the dense reference, 0.82 for the full sparse solve and 1.05 for the elimination solve, against reference values of 3 and 1; all three sit below their reference because the fit spans the overhead-dominated small sizes. Between the two largest sizes alone the local slopes are 2.91, 1.21 and 1.25, close to the cubic growth of dense LU for the reference and slightly above linear for the two structured solves. At 1,024 racks the dense reference takes 558 ms and the full sparse solve 27.3 ms, a ratio of 20.4 in the plotted repeat; over the five repeats the integer part of that ratio was 21, 23, 21, 20, 21, so by Eq. (25) the full sparse solve is at least 20× (20–23× over five repeats) faster. The interface-elimination solve returns the same state but is slower than the full sparse solve at every size, by a factor of 18.0 at 1,024 racks; it is faster than the dense reference only at 1,024 racks (490 against 558 ms).

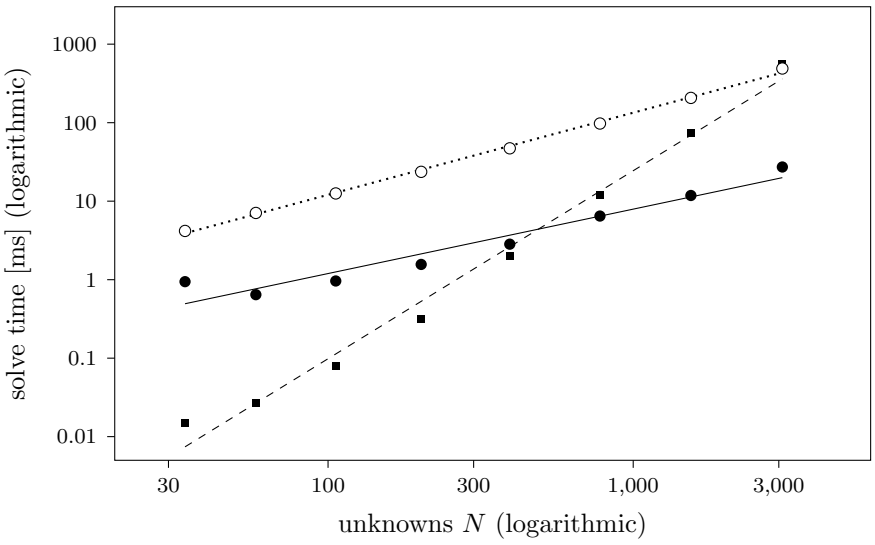


Figure 5: Measured solve time against the number of unknowns, both axes logarithmic, so a power law is a straight line whose slope is its exponent. Each marker is the fastest of five timed solves at one size, from the repeat of the protocol that gave the smallest speed bound; each line is the least-squares fit over all eight sizes. The dense reference starts far below the two structured solves, crosses the full sparse solve between 128 and 256 racks, and at 1,024 racks takes 558 ms against 27.3 ms.

Table 4: Measured solve times at each building size, the fastest of five timed solves per entry, from the repeat of the protocol that gave the smallest speed bound. The last column is the speed ratio of the full sparse solve over the dense reference; below one, the dense reference is faster.

Racks n	Unknowns N	Dense [ms]	Full sparse [ms]	Elimination [ms]	Dense / full sparse
8	34	0.0150	0.944	4.17	0.0159
16	58	0.0266	0.644	7.09	0.0413
32	106	0.0804	0.963	12.6	0.0835
64	202	0.317	1.56	23.6	0.203
128	394	2.01	2.83	47.2	0.708
256	778	12.2	6.46	97.5	1.88
512	1,546	75.0	11.8	207	6.35
1,024	3,082	558	27.3	490	20.4

5 Discussion

The mechanism behind the result is the one Parter (1961) described. The network is a tree, a rack's unknowns appear in no relation of another rack, and so eliminating them creates no coupling that was not already there. A sparse factorization that is free to choose its elimination order finds that structure without being told, which is why the full sparse solve, and not the explicit elimination, is the fastest of the three at every size where either beats the dense reference. The explicit elimination pays twice for its bookkeeping: it factorizes each of the 1,025 blocks separately, and it forms and factorizes the interface system as a dense matrix, whose cost grows as m^3: at 1,024 racks the blocks contribute less than 10^{-5} of the operation count in the denominator of Eq. (23), and the dense interface system all the rest.

The results stop at several limits. First, the times come from one laptop; they would differ on other hardware and under other load, and only the exponents and the bound are meant to transfer, not the milliseconds. Second, the dense reference solves a random matrix of the right size, not the network's own matrix. The cost of dense LU depends on the matrix size and hardly at all on its entries, but it remains a stand-in for the cost of a dense solver applied to this network. Third, the comparison is uneven in the dense reference's favor: its time covers the factorization and solve only, while the structured solves' times include assembling the relations and checking convergence. The bound is therefore conservative. Fourth, the fitted exponents mix the overhead-dominated small sizes with the large ones and understate the asymptotic rates; the local slopes between the two largest sizes are descriptive, from two points each, and carry no measure of their own spread. Fifth, the operation count is a cost model, not a measurement: it uses the average block size and treats the interface as dense. Sixth, the relations are linear, so every solve takes one Newton step; a nonlinear model multiplies each solve's cost by its number of steps. Seventh, only a steady-state tree was tested: air recirculation between racks, shared plenums or rows and aisles add couplings that enlarge the interface. Eighth, the largest system has 3,082 unknowns, memory was not measured, and nothing ran in parallel, although the independent blocks of the elimination solve are what a parallel solve would distribute. Finally, exploiting sparsity is standard practice, and any competent sparse solver given this system would scale as the full sparse solve does here; what the network description contributes is that the sparsity pattern is the building's own structure, with no separate step to extract it.

6 Conclusion

This report asked how the cost of solving a rack-resolved thermal network of a data center building grows with the number of racks when the solve follows the network's tree structure, against a dense factorization of a system of the same size. A building of four rooms, with every heat path sized for the heat it carries, was solved at eight sizes from 8 to 1,024 racks. The answer does not depend on how the elimination is organized: the full sparse solve and the interface-elimination solve agree to better than 10^{-11} relative everywhere, and every solved building passes its physical checks. The cost does. Eliminating the rack blocks saves a factor of operations that rises to 26.56 and tends to 27. Measured, the full sparse solve grows as $N^{0.82}$ and the dense reference as $N^{2.40}$ over the sweep, with local slopes of 1.21 and 2.91 at the largest sizes, and at 1,024 racks the full sparse solve is at least 20× (20–23× over five repeats) faster, on one laptop and against a dense reference that is timed on its linear algebra alone. The dense reference remains the faster choice at 128 racks and below. The next steps are the cases this report leaves out: nonlinear relations, networks with recirculation that are not trees, and systems large enough for memory and parallel elimination to

matter.

A Reproduction

```
cd docs/terra_graph_engine/case_studies/level1_building/graph_structure_scaling
PYTHONPATH=../../../../../src python run.py --out <directory>
PYTHONPATH=../../../../../src python make_report.py --pdf
OPENBLAS_NUM_THREADS=1 PYTHONPATH=../../../../../src python run.py --retime
cd ../../../../..
python -m pytest -q tests/plugins/datacenter/test_graph_structure_scaling_case_study.py
```

`run.py` builds and solves the building at the eight sizes and writes `results.json`, and copies the committed `timings.json` beside it; it never re-times. `make_report.py` reads only those two files and writes `report.tex`. The third command re-measures the timings under the protocol of Section 3.6 and rewrites the study's own `timings.json`; it refuses any other output directory. `results.json` is deterministic on a given machine; the round-off-level balances it records differ in their trailing digits between BLAS kernels, which is why this report states them as bounds. The dense reference's matrix comes from a generator seeded with 0. The timings were taken with Python 3.12.4 and NumPy 2.4.6. The test in the last line regenerates the results, compares them with the committed file and rebuilds the tex byte for byte.

B Data and provenance

No measured data enter this report; the building and its loads are constructed from the inputs of Table 1. The solve times in `timings.json` were measured on 2026-09-17 on a development laptop (`Windows-11-10.0.26200-SP0`, processor `Intel64 Family 6 Model 140 Stepping 1, GenuineIntel`) with a single BLAS thread. All five repeats of the protocol started in a quiet window, with no other Python, Cargo or LaTeX process running and a processor load between 7.6 and 11.6 % at the check. Conductances follow the sizing rule of Eqs. (11)–(14). The model at commit `5be377ed` used $G = 10$, 50 and 100 W/K (rack to room, room to building, building to chilled water) at every size; its figures are not comparable. The study leaves two open problems: the scaling of a network that is not a tree, where the interface grows faster than the number of racks, and a timing protocol whose bound transfers across machines rather than holding on the one it was measured on.

References

[Anderson et al. 1999] Anderson, E., Bai, Z., Bischof, C., Blackford, L. S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Sorensen, D. (1999). LAPACK Users' Guide, third edition. Society for Industrial and Applied Mathematics, Philadelphia, PA. doi:10.1137/1.9780898719604.

[Crawley et al. 2001] Crawley, D. B., Lawrie, L. K., Winkelmann, F. C., Buhl, W. F., Huang, Y. J., Pedersen, C. O., Strand, R. K., Liesen, R. J., Fisher, D. E., Witte, M. J., Glazer, J. (2001). EnergyPlus: creating a new-generation building energy simulation program. Energy and Buildings 33(4), 319–331. doi:10.1016/S0378-7788(00)00114-6.

[Davis 2006] Davis, T. A. (2006). Direct Methods for Sparse Linear Systems. Society for Industrial and Applied Mathematics, Philadelphia, PA. doi:10.1137/1.9780898718881.

[Demmel et al. 1999] Demmel, J. W., Eisenstat, S. C., Gilbert, J. R., Li, X. S., Liu, J. W. H. (1999). A supernodal approach to sparse partial pivoting. SIAM Journal on Matrix Analysis and Applications 20(3), 720–755. doi:10.1137/S0895479895291765.

[George 1973] George, A. (1973). Nested dissection of a regular finite element mesh. SIAM Journal on Numerical Analysis 10(2), 345–363. doi:10.1137/0710032.

[Harris et al. 2020] Harris, C. R., Millman, K. J., van der Walt, S. J., et al. (2020). Array programming with NumPy. Nature 585(7825), 357–362. doi:10.1038/s41586-020-2649-2.

[Masanet et al. 2020] Masanet, E., Shehabi, A., Lei, N., Smith, S., Koomey, J. (2020). Recalibrating global data center energy-use estimates. Science 367(6481), 984–986. doi:10.1126/science.aba3758.

[Parter 1961] Parter, S. (1961). The use of linear graphs in Gauss elimination. SIAM Review 3(2), 119–130. doi:10.1137/1003021.

[Saad 2003] Saad, Y. (2003). Iterative Methods for Sparse Linear Systems, second edition. Society for Industrial and Applied Mathematics, Philadelphia, PA. doi:10.1137/1.9780898718003.

[Tang et al. 2008] Tang, Q., Gupta, S. K. S., Varsamopoulos, G. (2008). Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: a cyber-physical approach. IEEE Transactions on Parallel and Distributed Systems 19(11), 1458–1472. doi:10.1109/TPDS.2008.111.

[Virtanen et al. 2020] Virtanen, P., Gommers, R., Oliphant, T. E., et al. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature Methods 17(3), 261–272. doi:10.1038/s41592-019-0686-2.

[Wetter et al. 2014] Wetter, M., Zuo, W., Nouidui, T. S., Pang, X. (2014). Modelica Buildings library. Journal of Building Performance Simulation 7(4), 253–270. doi:10.1080/19401493.2013.765506.