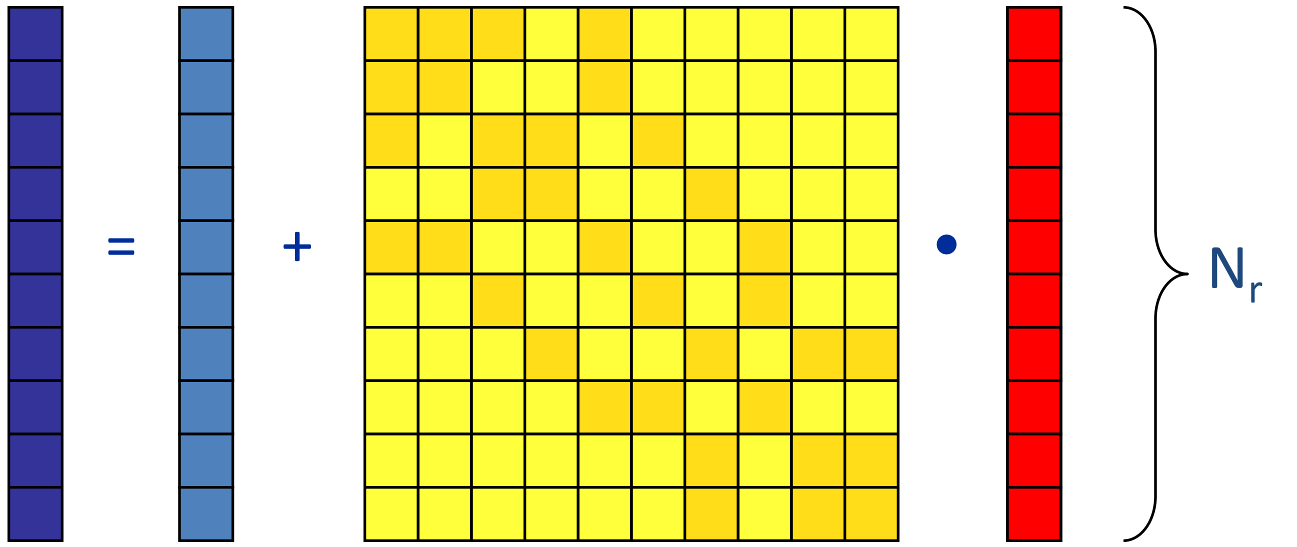
Some Theory on Sparse Matrix-Vector Multiplication (SpMV)

Sparse Matrix Vector Multiplication (SpMV)

- Key ingredient in some matrix diagonalization algorithms
  - Lanczos, Davidson, Jacobi-Davidson
- Store only N_{nz} nonzero elements of matrix and RHS, LHS vectors with N_r (number of matrix rows) entries
- "Sparse": $N_{nz} \sim N_r$
- Average number of nonzeros per row: $N_{nzr} = N_{nz}/N_r$



General case: some indirect addressing required!

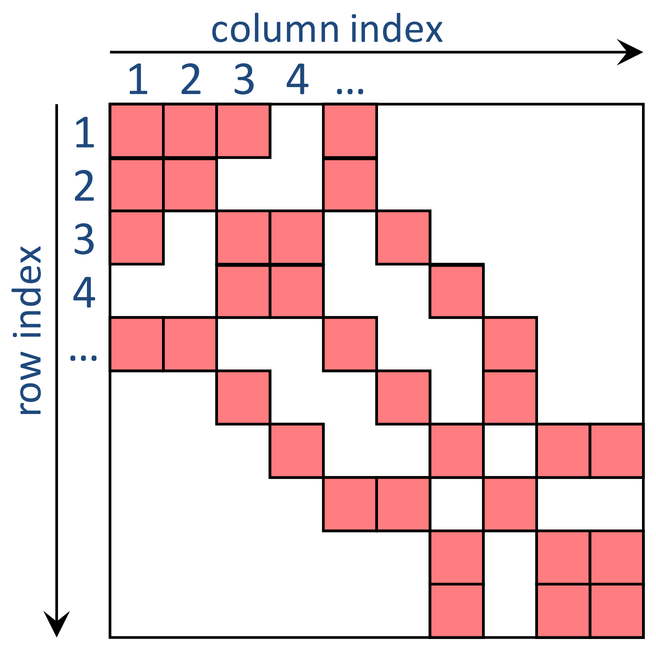
SpMVM characteristics

- For large problems, SpMV is inevitably memory-bound
  - Intra-socket saturation effect on modern multicores

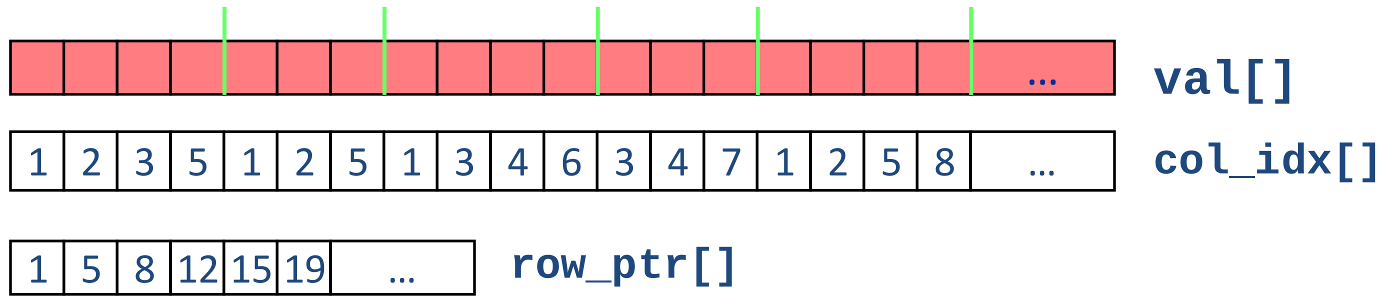
- SpMV is easily parallelizable in shared and distributed memory
  - Load balancing
  - Communication overhead

- Data storage format is crucial for performance properties
  - Most useful general format on CPUs:
    Compressed Row Storage (CRS)
  - Depending on compute architecture

CRS matrix storage scheme



- **`val[]`** stores all the nonzeros (length N_{nz})
- **`col_idx[]`** stores the column index of each nonzero (length N_{nz})
- **`row_ptr[]`** stores the starting index of each new row in **`val[]`** (length: N_r)



1	2	3	5	1	2	5	1	3	4	6	3	4	7	1	2	5	8	…

1	5	8	12	15	19	…

Case study: Sparse matrix-vector multiply

- Strongly memory-bound for large data sets
  - Streaming, with partially indirect access:

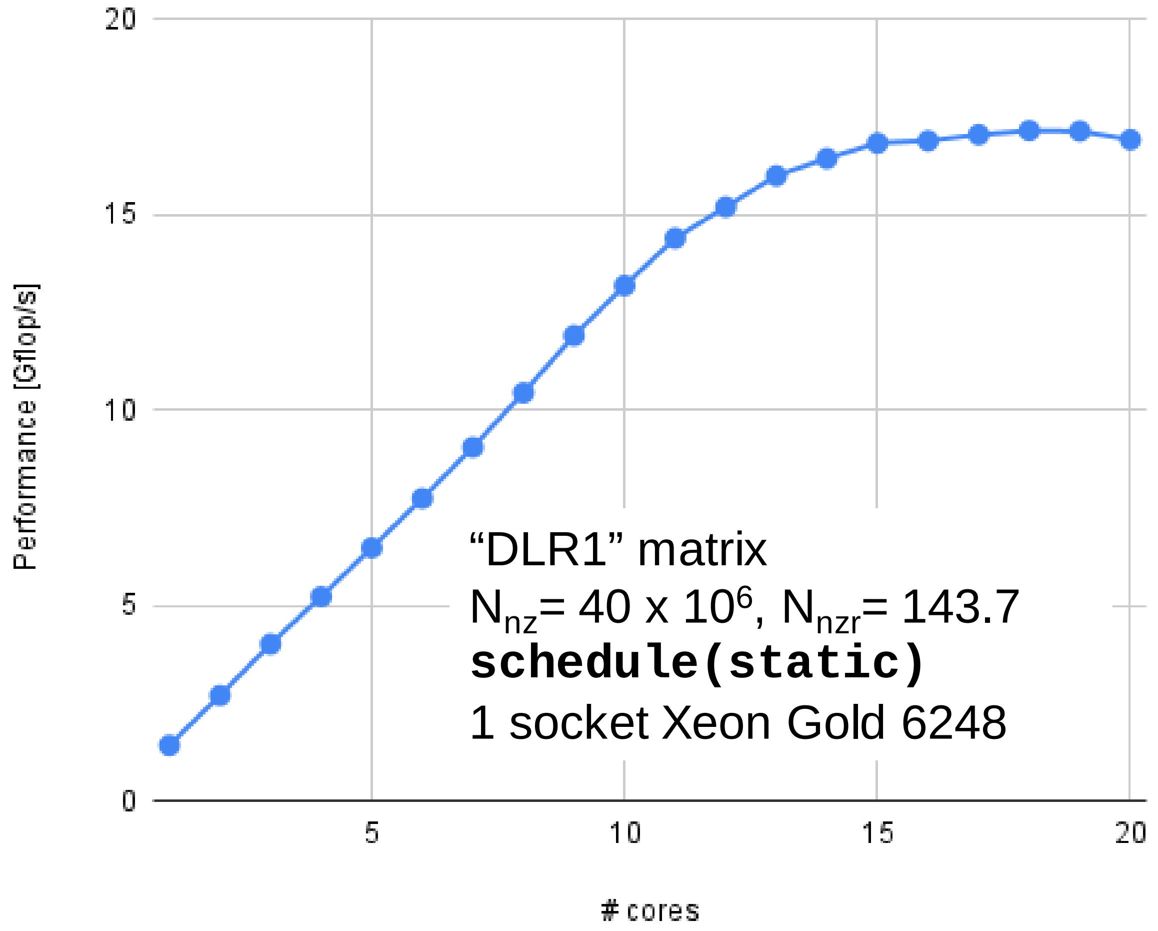
```
!$OMP parallel do schedule(???)
do i = 1,Nr
 do j = row_ptr(i), row_ptr(i+1) - 1
  C(i) = C(i) + val(j) * B(col_idx(j))
 enddo
enddo
!$OMP end parallel do
```

  - Usually many spMVMs required to solve a problem

- Now let’s look at some performance measurements…

Performance characteristics

- Strongly memory-bound for large data sets → saturating performance across cores on the chip
- Is the observed performance good or bad?
- Is there a "light speed" for SpMV?
- Optimization?



"DLR1" matrix
N_{nz}= 40 x 10^6, N_{nzr}= 143.7
`schedule(static)`
1 socket Xeon Gold 6248

Deriving useful upper performance limits

Roofline model delivers upper performance limit P for a loop:

$$P = \min\left(P_{max}, \frac{b_S}{B_C}\right)$$

- b_S: max. memory bandwidth
- B_C: code balance in byte/flop (inverse of computational intensity)
- P_{max}: max. theoretical performance of loop, assuming data is in the L1 cache
  - SpMV: `C(i) = C(i) + val(j) * B(col_idx(j))`
  - 4 loads, 1 store, 1 multiply, 1 add → load bound in L1 → insignificant! (?)

SpMV node performance model – CRS (1)

```
do i = 1, N_r
 do j = row_ptr(i), row_ptr(i+1) - 1
  C(i) = C(i) + val(j) * B(col_idx(j))
 enddo
enddo
```

```
real*8      val(N_nz)
integer*4 col_idx(N_nz)
integer*4 row_ptr(N_r)
real*8      C(N_r)
real*8      B(N_c)
```

Min. load traffic [B]: $(8+4)\,N_{nz} + (4+8)N_r + 8\,N_c$

Min. store traffic [B]: $8\,N_r$

Total FLOP count [F]: $2\,N_{nz}$

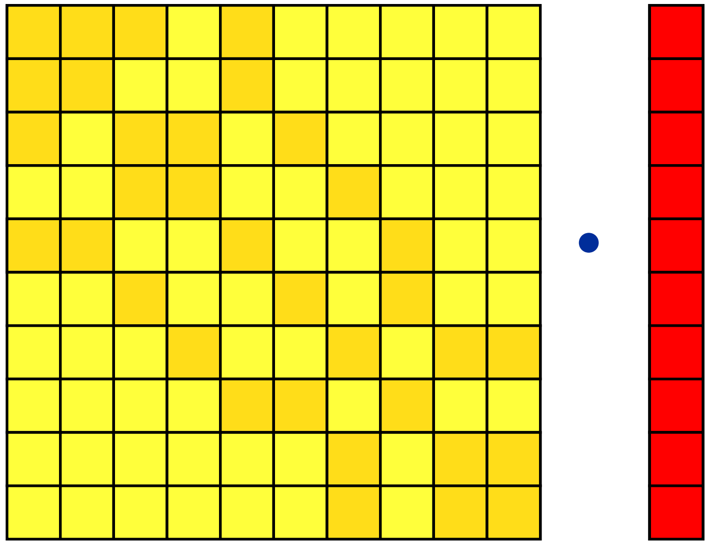
$$B_{C,min} = \frac{12\,N_{nz} + 20\,N_r + 8\,N_c}{2\,N_{nz}}\,\frac{B}{F} = \frac{12 + 20/N_{nzr} + 8/N_{nzc}}{2}\,\frac{B}{F}$$

Nonzeros per row ($N_{nzr} = {}^{N_{nz}}/_{N_r}$) or column ($N_{nzc} = {}^{N_{nz}}/_{N_c}$)

Lower bound for code balance: $B_C^{min} \geq 6\,\frac{\mathrm{B}}{\mathrm{F}}$ $\rightarrow I_{\max} \leq \frac{1}{6}\,\frac{\mathrm{F}}{\mathrm{B}}$

SpMV node performance model – CRS (2)

```
do i = 1, N_r
 do j = row_ptr(i), row_ptr(i+1) - 1
  C(i) = C(i) + val(j) * B(col_idx(j))
 enddo
enddo
```



$$B_C^{min} = \frac{12 + 20/N_{nzr} + 8/N_{nzc}}{2} \frac{B}{F}$$

$$B_C(\alpha) = \frac{12 + 20/N_{nzr} + \boxed{8\,\alpha}}{2} \frac{B}{F}$$

Parameter (α) quantifies additional traffic for **B(:)** (irregular access):

$$\alpha \geq {}^{1}/_{N_{nzc}}$$

$$\alpha N_{nzc} \geq 1$$

Consider square matrices: $N_{nzc} = N_{nzr}$ and $N_c = N_r$

Note: $B_C\left({}^{1}/_{N_{nzr}}\right) = B_{C,min}$

The “α effect”

DP CRS code balance

$$B_C(\alpha) = \frac{12 + 20/N_{nzr} + 8\,\alpha}{2}\,\frac{B}{F} = \left(6 + 4\,\alpha + \frac{10}{N_{nzr}}\right)\frac{B}{F}$$

- α quantifies the traffic for loading the RHS
  - $\alpha = 0$ → RHS is in cache
  - $\alpha = 1/N_{nzr}$ → RHS loaded once
  - $\alpha = 1$ → no cache
  - $\alpha > 1$ → Houston, we have a problem!
- “Target” performance = b_S/B_c
- Caveat: Maximum memory BW may not be achieved with spMVM (see later)

Can we predict α?

- Not in general
- Simple cases (banded, block-structured): Similar to layer condition analysis

→ But we can learn more by measuring the actual code balance, B_C^{meas}

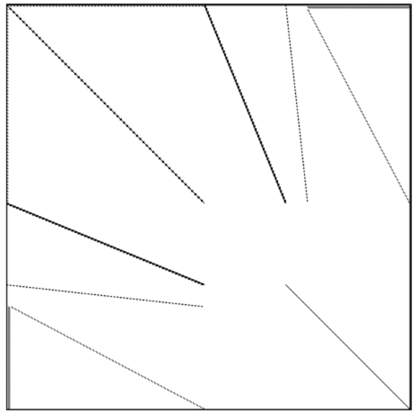
Measure B$_c$ (RHS extra traffic quantification)

$$B_C\ (\alpha) = \left(6{+}4\alpha{+}\frac{10}{N_{nzr}}\right)\frac{\mathrm{B}}{\mathrm{F}} = \frac{V_{meas}}{N_{nz}\cdot 2\ \mathrm{F}} = B_C^{meas}$$

V_{meas} is the measured overall memory data traffic (using, e.g., likwid-perfctr)

Example: kkt_power matrix from the UoF collection
on one Intel Sandy Bridge socket

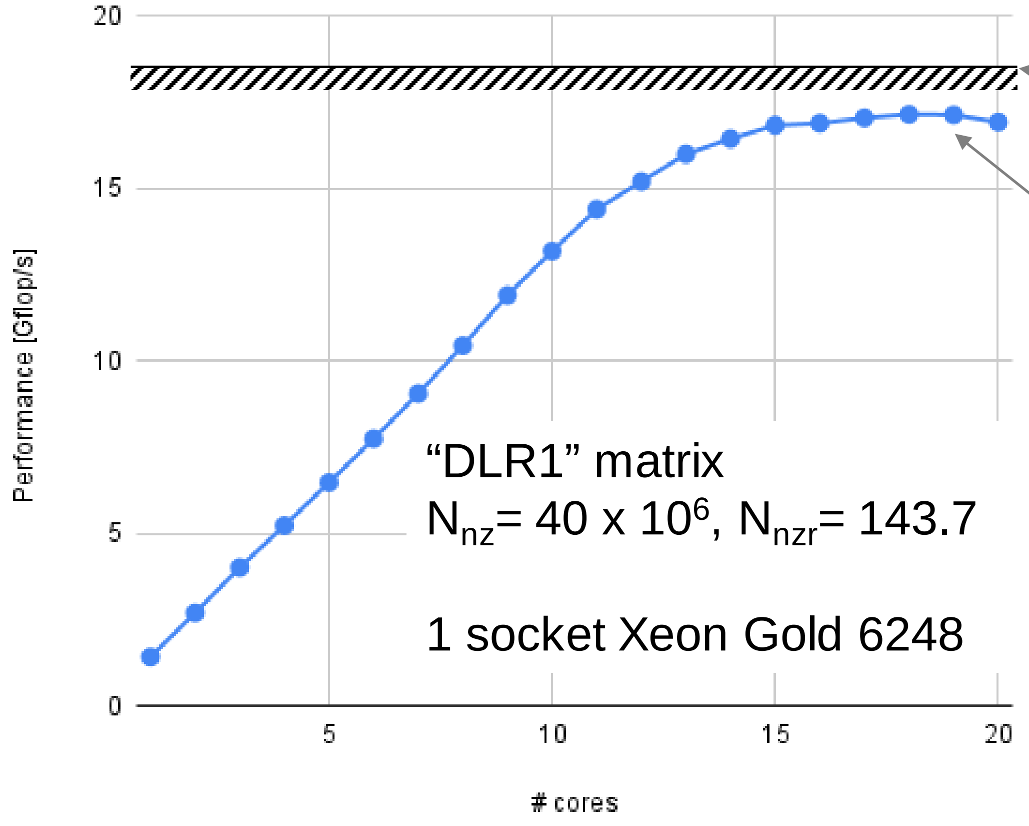
- $N_{nz} = 14.6\cdot 10^6, N_{nzr} = 7.1 \rightarrow B_C^{min} = 7.97\,\frac{\mathrm{B}}{\mathrm{F}}$
- $V_{meas} \approx 258\ \mathrm{MB} \rightarrow B_C^{meas} = 8.83\,\frac{\mathrm{B}}{\mathrm{F}}$



$$\frac{B_C^{meas}}{B_C^{min}} = 1.11$$

11% extra traffic → optimization potential!

Now back to the start…



"DLR1" matrix
N_{nz}= 40 x 10^6, N_{nzr}= 143.7

1 socket Xeon Gold 6248

- $b_S = 110\,\mathrm{GB/s}\,,\quad B_C^{best} = 6\,\mathrm{B/F}$
- Maximum CRS SpMV performance:

$$P_{limit} = 18.3\,\mathrm{GF/s}$$

- DLR1 has $N_{nzr} = 143.7$

  → light speed for DLR1:

  $B_C^{min} = 6.13\,\frac{\mathrm{B}}{\mathrm{F}} \rightarrow P \leq 18.0\,\frac{\mathrm{GF}}{\mathrm{s}}$

  → Measured maximum: 17.2 GF/s

  → DLR1 causes almost minimum CRS code balance (as expected)